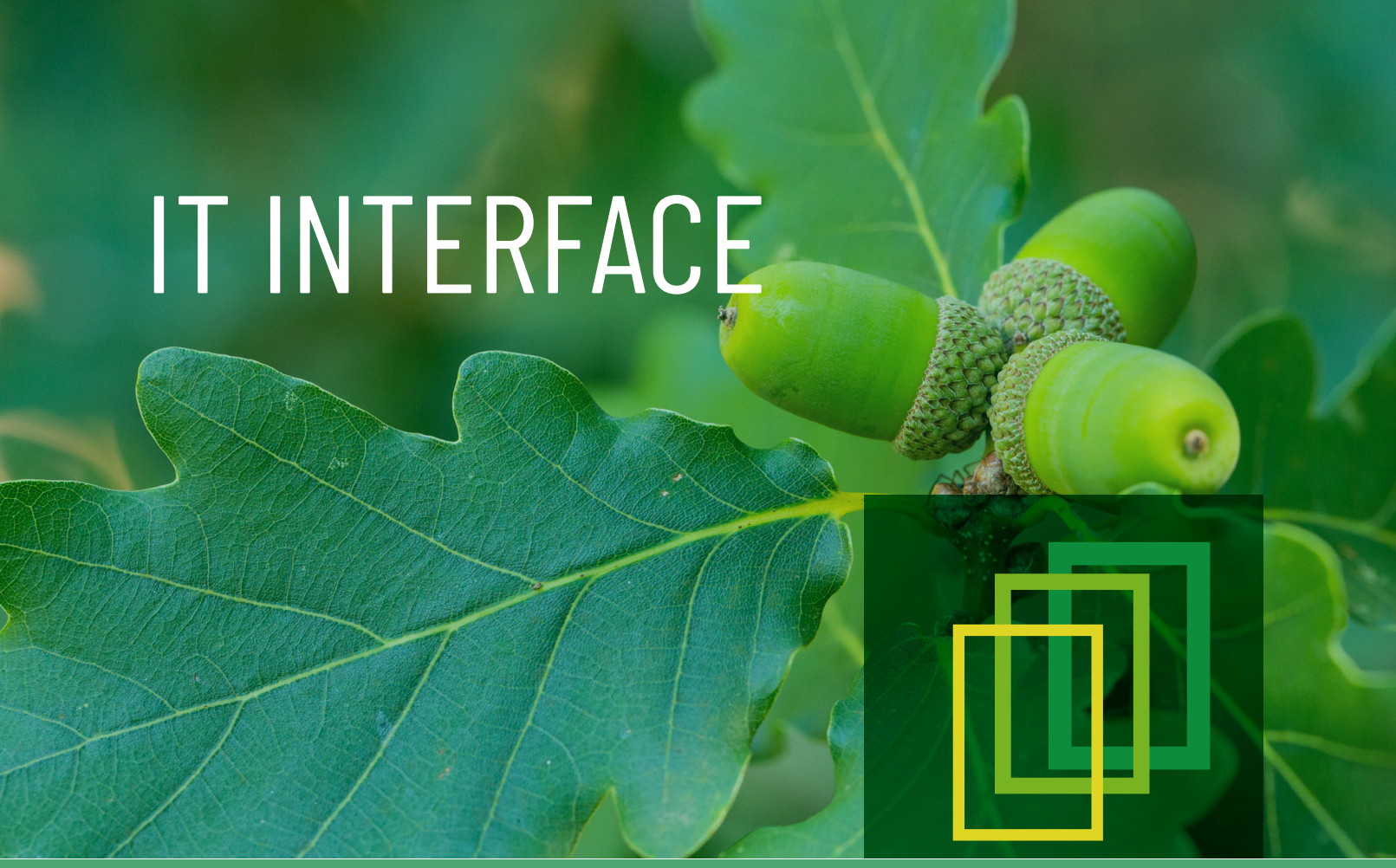


IT INTERFACE

DELIVERABLE D4.2
IT-INTERFACE ENGLISH

LIFE climate value chains

Informations about LIFE

The LIFE program forms the basis for measures to promote environmental and climate protection by the European Union. The aim of the LIFE program is to establish environmentally friendly, innovative products, processes and services as well as best practice in Europe and to further develop the corresponding policy and administrative practice. The program forms a bridge between research and implementation on a large scale.

Das Programm LIFE bildet die Grundlage für Maßnahmen zur Förderung des Umwelt- und Klimaschutzes durch die Europäische Union. Ziel des Programms LIFE ist es, umweltfreundliche, innovative Produkte, Verfahren und Dienstleistungen sowie Best Practice in Europa zu etablieren und die entsprechende Politik und Verwaltungspraxis weiterzuentwickeln. Das Programm bildet eine Brücke zwischen der Forschung und der Umsetzung im großen Maßstab.

Document informations

Doc. ref. No.: LIFE_CVC_C4_2

In coordination with a European Working Group within the frame of a projekt fundet by the European LIFE programme.

Autors: Dr. Gabriele Bruckner and Dr. Philipp Strohmeier (HVH, Germany)

Co-Autors: Luca Galeasso and Erik Dalmas (Envi park Turin, Italy). Bernard Likar and Helena Cvenkel (Wood industry clus-ter, BSC Kranj). Erich Rainer (Ing, HVH, Austria).

Publisher of the document:
Holz von Hier gemeinnützige GmbH. Neuenreuth 24, DE-95473 Creußen Germany; www.holz-von-hier.eu

This guide is available in printed and digital form.

The digital guide is continuously updated with new developments or new practical examples, while the print version is published at regular intervals. The evironmental Labels LOW CARBON TIMBER© and HOLZ VON HIER© are trademarks of the non-profit organisation Holz von Hier.

The contents of this publication are the sole responsibility of Holz von Hier and do not necessarily reflect the opinion of the European Union

The project LIFE Climate Value Chains received funding from the LIFE program of the European commission.

Contents

IT INTERFACES

1 / IT Interfaces

1.1 / Introduction

Two types of IT interfaces were implemented in the project. Only one interface was originally planned in the application, but the latest developments and requirements from the European Union regarding the EUDR made it necessary to develop a second type of interface.

(1) IT interface value chain

The first interface was developed to optimize the HVH-LCT verification management processes between companies in the supply chain. Since larger companies purchase such a large number of raw material deliveries or market products to a large number of customers, it is hardly possible to handle the transaction processes manually via the HVH-LCT electronic controlling system, as is customary. The interface enables the automated execution of HVH-LCT certifications of deliveries or product batches, so that larger companies can also use the certification system. This means that even large quantities of wood product flows can be recorded, influenced and made climate-friendly.

With the help of the project, a modern RESTful API interface tailored to timber supply chains was created. Such modern interfaces are also used by large companies such as Amazon for the complex organization of goods distribution.

(2) EUDR IT interface

The second IT interface is an automated connection to the European Commission's "TRACES" platform, the central platform for the implementation of the new European regulation on deforestation-free supply chains (EUDR). The platform is used for the submission of due diligence declarations by companies in supply chains whose raw materials originate from areas that could potentially be affected by deforestation.

Since timber processing companies are affected by the EUDR and are obliged to submit such due diligence declarations, a corresponding interface between the digital controlling system of the HVH/LCT and the EU platform was to be developed in order to be able to provide data and information required by the EUDR via HVH/LCT.

Such an interface was very important in order for the HVH/LCT to be able to offer support to companies in the timber industry within the framework of the EUDR. Otherwise, there is a risk that companies will save costs and effort for voluntary systems such as HVH/LCT and perceive the new, mandatory EUDR as an additional burden.

1.2 / LCT-RESTful-API-base

1.2.1 / Overwiev

The IT interface was specially developed to provide customers with seamless and efficient integration with the LCT/HVH system. At its heart is a specially designed RESTful API that supports both JSON-based data transfer and server-side transaction execution. This API makes it possible to automate business processes and connect existing IT systems directly to the LCT/HVH platform without having to rely on the graphical user interface.

The architecture of the RESTful API was designed with a focus on scalability, security and performance. Among other things, it supports HTTPS-secured communication, token-based authentication and granular access rights (precise control of access rights).

Thanks to support for synchronous and asynchronous data handling, both real-time data and complex batch operations can be processed. Comprehensive exception handling with standardized error feedback enables rapid problem analysis and increases operational reliability. The IT interface thus creates the basis for future-proof and modular extensions that can be tailored precisely to the individual needs of supply chain participants.

1.2.2 / RESTful-API-base

A programming interface, an "API", defines the rules that users must follow to communicate with other software systems. APIs are used so that applications can communicate programmatically and without problems. APIs are always geared towards the respective defined main aspect or main aspects of communication. Most API interfaces are geared towards "clients" and "resources". Clients are users who want to access information. The client can be a person or a software system that uses the API. Resources in the sense of an IT interface are information that various applications make available to their clients. The "resources" can be images, videos, text, numbers or any other type of data. Organizations use APIs to share resources and provide web services while maintaining security, control and authentication. APIs also help to define in detail and differentiate which client is allowed to access certain or which internal resources.

A Representational State Transfer (REST) based interface, a "RESTful API", is an interface that allows two computer systems to exchange information securely over the Internet. Most business applications need to communicate with other internal and external applications to perform various tasks. RESTful APIs support this information exchange by following secure, reliable and efficient software communication standards. REST-based IT architecture is used today to support high-performance reliable communication between two IT systems according to the specified level, while maintaining the highest possible IT security.

Some principles of the RESTful API interface are briefly described below:

Uniformity. The basis of a RESTful web service is that the server transfers information in a standard format. The formatted "resource" is referred to as a representation in REST. This format may differ from the internal representation of the resource on the server application. Uniformity imposes architectural constraints, such as the following: Requests should identify resources, they do so by using a uniform resource identifier. Clients have sufficient information in the resource representation to modify or delete the resource if they wish. The server fulfills this condition by sending metadata that describes the resource in more detail. Clients receive information about the further processing of the representation. The server accomplishes this by sending self-describing messages that contain metadata about how the client can best use it. Clients receive information about any other related resources they need to complete a task.

Statelessness. In the REST architecture, "statelessness" refers to a communication method in which the server completes each client request individually and independently of all previous requests. This minimizes errors, especially in complex contexts such as supply chains, and makes it possible to control them individually in the first place. Only in this way can individual supply chains be monitored at all. Clients can request resources in any order and each request is stateless or isolated from other requests. The REST API design must be such that the server can fully understand and process the request every time.

Layered model. In the layered architecture of the RESTful API, the "client" can connect to other authorized intermediaries between the client and the ser-

ver and still receive responses from the server. The servers can also forward requests to other servers. The RESTful API is designed to run on multiple servers with multiple layers such as security, application and business logic that work together to fulfill client requests. However, these layers remain invisible to any "client".

Code on-demand. In the REST IT architecture style, servers can temporarily extend or adapt the client functionality by transferring software programming code to the client. If an error occurs, for example, this is reported back almost immediately by the interface in real time.

1.2.3 / Advantages

The main advantages of the RESTful API over conventional APIs are scalability, flexibility and independence.

a) Scalability

Systems that implement REST APIs can scale efficiently because REST optimizes the interactions between client and server. Statelessness removes server load because the server does not need to store information about previous client requests. Well-managed caching removes some client-server interactions partially or completely. All of these features support scalability without causing communication bottlenecks that would affect performance.

Scalability is the ability of a system to change size. Good scalability allows systems to grow well or makes this possible in the first place. In IT, scalability means the ability of a system of hardware and software to increase performance by adding "resources", such as additional processes or hardware, in defined areas. This possibility should be kept open for LCT/HVH and scalability was one of the important aspects and programming requirements for LCT/HVH.

"Statelessness" refers to the property of an IT communication protocol or a distributed system to treat requests as independent transactions. The communication participant does not save a protocol state and therefore cannot link requests from the same client. Another advantage of statelessness is the reduced complexity, as no context or session information needs to be stored and managed. This makes load balancing and scalability easier to implement.

"Load balancing" is used in IT processes to distribute extensive calculations or large volumes of requests across several systems working in parallel with the aim of making their overall processing more efficient. If the individual processes are largely independent of each other, the architecture form of the computer cluster is suitable, in which the processes are distributed across a certain number of similar servers, a so-called "server farm". This approach is often used for larger web applications with many users. This option should be kept open for LCT/HVH.

If, on the other hand, it is a single, very complex process, an attempt can be made to split the task and then merge the results, such as when balancing a very large number of bookings. This option should also be kept open for LCT/HVH.

b) Flexibility

RESTful IT services support the total separation between client and server. RESTful IT services simplify and decouple various server components so that each part can develop independently. Platform or technology changes to the server application do not affect the client application. The ability to split application functions into layers even increases flexibility. For example, developers can make changes to the database layer without having to rewrite the application logic. This was one of the most important aspects for LCT/HVH in order to be able to continue to flexibly adapt the LCT-HVH system to requirements such as those imposed by new EU regulations for supply chains.

c) Independence

REST APIs are independent of the technology used. It is possible to write both client and server applications in different programming languages without affecting the API design. It is also possible to change the underlying technology on both sides without affecting communication. This was one of the important aspects for LCT/HVH, because supply chain networks in particular have various users with diverse internal IT systems, even of different modernity.

Most IT-supported internal IT systems are designed by the programming forms or program providers in such a way that their programming languages and codes and their systems are used, thus creating a

bond between the customer and the programming house. RESTful APIs allow users to continue using their own IT systems as they are used to without having to make major and enormously costly changes. Although certain docking adaptations to the RESTful API are necessary, such as output formats, a RESTful API reduces the effort here enormously, as it makes communication between very different IT systems possible in the first place, possibly at a very different level of modernity.

1.2.4 / JSON

JSON is a completely independent data format with no connection to the JavaScript language. At the same time, JSON is the ideal format for exchanging data between systems due to its very simple structure and encoding in the Unicode character set. It is always exchanged between applications as a whole.

JSON (JavaScript Object Notation) is a data exchange format that provides a standardized and efficient way for data to be exchanged between different systems. Thanks to its simplicity, flexibility and compatibility with common programming languages, JSON has become one of the most modern IT technologies. JSON is a text-based format for storing and exchanging data that is both human- and machine-readable. Although JSON has its roots in JavaScript, it has grown into a very powerful data format that simplifies data exchange across different platforms and programming languages.

JSON is a popular data format that is now commonly used for data transfer between a server and a web application. Since JSON is text-based, it can be easily read by humans and understood by computers. The language-independent nature of JSON makes it an ideal format for exchanging data between different programming languages and platforms. There are now many databases in which data is stored and exchanged in JSON. Beyond web development, JSON is often used in an application or IT system to store and manage configuration settings. For example, configuration files written in JSON format can contain important information, such as database connection details, API keys or user settings. Storing configuration data in simple, easy-to-read and easy-to-parse JSON files makes it possible to change application settings without the need for code changes. JSON is a flexible format for data interchange that is widely supported across modern programming languages and software systems. It is text-based and lightweight, and has a simple, easy-to-parse data format, meaning that no additional code is required to understand and interpret the data provided. JSON has gained traction in API programming because it enables faster data exchange and quicker results. It also has the advantage of providing easy access to open-source and NoSQL document databases, which store data in JSON format and require no additional processing when exchanging data.

1.2.5 / Tokens

However, using tokens requires a high level of programming knowledge and is therefore not used for every system. Token-based authentication is a protocol that allows users to verify their identity and in return receive a unique “access token”. While the security token is valid, users have access to the application for which the token was issued. They do not have to re-enter their credentials each time they access the same application or “resource” protected by the token.

Tokens provide an additional layer of security. Authentication tokens work like a stamped ticket. The user has access as long as the token is valid. As soon as the user logs off or exits an application, the token is canceled. Token-based authentication differs from traditional password- or server-based authentication

Tokens facilitate or enable external monitoring of digital interfaces. In the case of the LCT/HVH REST API, this also allows the system certifiers (TÜV) to closely monitor every action and transaction within and via the interface. This is an extremely important aspect in terms of external monitoring of the LCT/HVH system, including (!) the interface.

1.2.6 / Practice test in the project

The functionality of the RESTful API interface described must ultimately be tested and tried out using practical examples. In the project, this was implemented using the example of a supply chain related to windows. Such a supply chain presents particular challenges with regard to certification with Holz von Hier, since, for example, window manufacturers pro-

cure raw materials on an order-by-order basis. Here, a large number of small orders are placed, rather than a small number of large orders for raw materials.

Likewise, certification of product deliveries to a large number of customers is necessary. While, for example, a larger timber construction company may build 50 houses per year that require certification, a window manufacturing company of the same size may have as many as 20,000 customer-related individual deliveries that would need to be certified. This is why an interface for automating the transaction processes is particularly important here.

The supply chain in the practical test comprises two nodes in succession and reflects the entire chain from the origin of the raw material to the end customer (forest – sawmill (and cantel manufacturer) – window manufacturer – customer). In this specific case, the window manufacturer's orders for raw materials for end customers are already referenced, which optimizes traceability but makes the programming of the interface function, which has to be carried out for each individual company, more complex. This has led to a significantly longer implementation period, but on the other hand, it documents and ensures the feasibility for further specific companies and applications, which are usually more straightforward.

1.2.7 / Further action

The programming required in the project mainly concerned the fundamental implementation of a RESTful API in the digital controlling system of Holz von Hier. This work is fundamental, but only necessary once. The basis for this was therefore created in the project. However, a specific use of this interface function by a specific company still requires an adaptation or addition to the individual circumstances and specifications of the IT system used by the company. These adaptations were and are not part of the project, as they serve specific individual companies and not the whole. Such individual adaptations are therefore to be borne by interested companies themselves after the end of the project.

1.3 / EUDR-HVH/LCT-IT

1.3.1 / SOAP based EUDR-HVH/LCT-IT-Interface

After programming the "LCT-HVH-RESTful-API-base" based on REST, the EUDR requirement was issued in the European Union, which is to be implemented in 2025. Since the European due diligence was based on a SOAP API, but the LCT-HVH interface was based on REST, an additional module had to be created for the LCT/HVH system to enable a connection to the SOAP-based EUDR system. This is the only way for companies in the LCT-HVH network to fulfill EUDR requirements via the LCT/HVH system as part of certification transactions.

SOAP and REST are two different approaches to API design. The SOAP approach is highly structured and uses the XML data format. REST is more flexible and allows applications to exchange data in different formats.

SOAP and REST are used to create APIs or communication points between different applications. Both describe rules and standards for how applications make, process and respond to data requests from other applications. Both use HTTP, the standardized internet protocol, to exchange information. Both support SSL/TLS for secure, encrypted communication.

However, SOAP is an older technology that requires a more rigid exchange of data between systems. REST was developed after SOAP and inherently addresses many shortcomings that SOAP still has. REST is therefore the more modern IT architecture today.

SOAP is the Simple Object Access Protocol, a messaging standard defined by the World Wide Web Consortium and its members. This is probably one of the main reasons why this format was chosen for the European EUDR platform.

1.3.2 / Differences between REST and SOAP

The following table shows an overview of the differences between the two API structures SOAP and REST (Tab. 1).

	SOAP	REST
What does it do?	SOAP is a protocol for communication between applications.	REST is an architectural style for designing communication interfaces.
Design	The SOAP API makes the process available.	The REST API makes the data available.
Transport-protocoll	SOAP is independent and can work with any transport protocol.	REST only works with HTTPS.
Data format	SOAP only supports XML data exchange.	REST supports XML, JSON, plain text, HTML.
Performance	SOAP messages are larger, which slows down communication..	REST offers faster performance due to smaller messages.
Scalability	SOAP is difficult to scale. The server maintains the state by storing all previous messages. .	REST is easy to scale. It is stateless, so each message is processed independently of previous messages.
Security	SOAP supports encryption with additional overhead.	REST supports encryption without performance impact.
Use	SOAP is useful in legacy applications and private APIs.	REST is useful in modern applications and public APIs.

Tab. 1) Overview of the differences between the two API structures.

SOAP is a protocol, while REST is an "IT architecture style". This leads to significant differences in the behavior of SOAP APIs and REST APIs.

Key features of SOAP are

1. SOAP always uses an XML data format.
2. SOAP is a protocol that defines rigid communication rules.
3. There are several associated standards such as Web Services Security (WS-Security), the addressing of web services (WS-Addressing) by specifying routing information as metadata.
4. WS-ReliableMessaging standardizes error handling in SOAP messaging.
5. The Web Services Description Language (WSDL) describes the scope and function of SOAP web services.
6. When sending a request to a SOAP, the HTTP request must be packaged in a "SOAP envelope." This is a data structure that modifies the underlying HTTP content based on SOAP request requirements. The envelope also allows you to send requests to SOAP web services using other transport protocols, such as TCP or Internet Control Message Protocol (ICMP). These "envelopes" harbor uncontrolled possibilities for error.

By comparison, REST is a software architecture style that imposes conditions on how an API works, which are required by more flexible applications. These include:

1. Sender and receiver are independent of each other in terms of technology, platform, programming language (client-server architecture).
2. The server can have multiple intermediaries that work together to process client requests, but they are invisible to the client (layered structure).
3. The API returns data in a standard format that is complete and fully usable (interface uniformity).
4. The REST API completes each new request independently of previous requests (stateless).
5. All REST API responses can be cached (cachable), which is important for external certification of the interface function.
6. The REST API response can include a code snippet if needed (code on demand).
7. Rest API responses are usually in JSON, but can also be in another data format.

1.3.3 / Basics SOAP

(1) Design

The SOAP API makes functions or operations available, while REST APIs are mostly data-driven. Modern applications such as mobile apps and hybrid applications work better with REST APIs. REST offers the scalability and flexibility to use applications with modern architecture patterns (e.g. microservices, containers, etc.). However, if you need to integrate or extend older systems, such as the IT systems in the EU (e.g. customs and others) that already have older IT solutions, it may be better to develop an interface with SOAP.

However, this usually requires users to make a much more extensive customization of existing internal IT than with REST or the LCT-HVH REST API. Therefore, in terms of adapting the LCT-HVH system to the EUDR, extensive customizations also had to be made in the LCT-HVH system itself.

(2) Flexibility

SOAP systems are rigid and only allow XML messaging between applications. The application server must also maintain the state of each client. This means that when processing a new request, it must remember all previous requests. REST is more flexible and allows applications to transfer data as plain text, HTML, XML and JSON. REST is also stateless, so the REST API handles each new request independently of previous requests.

Flexibility was a very important requirement for the first LCT-HVH interface (see chapter x), which is why REST was the solution of choice here.

(3) Performance

SOAP messages are larger and more complex, which means that they can be transmitted and processed more slowly. This can increase page loading times. REST is faster and more efficient than SOAP due to the smaller message sizes of REST. REST responses can also be cached, allowing the server to store frequently retrieved data in a cache for even faster page loading times.

Speed and performance were very important for the real-time recording of supply chains in LCT/HVH, which is why the REST-based solution was the first choice for the first interface. Since deliveries can also be bundled in the EUDR implementation, sometimes even once a year, speed is not an essential requirement here.

(4) Scalability

The SOAP protocol requires applications to store state between requests, which increases bandwidth and memory requirements. As a result, applications become expensive and scaling is difficult. In contrast to SOAP, REST enables a stateless and multi-layer architecture, making it more scalable. For example, the application server can forward the request to other servers or leave it to an intermediary (e.g. a content delivery network) to process.

Scalability is always an important criterion for IT solutions for LCT/HVH, which is why the REST-based solution was the first choice for the interface. Since the EUDR system will probably not require any scaling and is intended to remain an isolated solution, the aspect of scalability was probably not an important criterion here. Many other EU-supported IT systems are also isolated solutions so far.

(5) Security

SOAP requires an additional WS security layer to work with HTTPS. WS-Security uses additional header content to ensure that only the specified process on the specified server reads the content of the SOAP message. This increases the communication effort and has a negative impact on performance. REST supports HTTPS without any additional effort.

REST APIs therefore offer more flexibility, which is required, for example, in networks or changing supply chains. In contrast, SOAP APIs are often used within individual companies or corporate structures or by public authorities to map internal company requirements.

The special solution of the new LCT-HVH-REST-API offers a modern comparable, if not higher, data security than conventional SOAP solutions by using various adds such as tokens and reprogramming.

(6) Reliability

SOAP has an integrated error handling logic and of-

fers more reliability. On the other hand, REST requires that you retry in case of communication failures.

This has been greatly optimized in the LCT-HVH REST API through comprehensive exception handling with well-known iterative adaptive and standardized error responses. This now enables very fast problem analysis and increases operational reliability. In the future, this can also be used to build AI solutions for problem analysis if desired and necessary.

(7) ACID compliance

SOAP has built-in compliance for atomicity, consistency, isolation, and durability (ACID). If you want to implement comparable requirements with REST, REST APIs may require additional software modules to enforce the status at the server or database level. This is the case with the new LCT-HVH REST API.

1.3.4 / Programming in the LCT-HVH system for EUDR

The EUDR programming includes the complete technical connection to the European Due Diligence SOAP API. This includes the extension of the existing MariaDB database structure with additional tables, foreign key relationships and indexes to meet the increased data integrity, consistency and processing speed requirements.

The server-side logic was implemented in the PHP Zend Framework, including dedicated transaction controllers for consistent data processing.

To optimize user interaction, a new, intuitive user interface was implemented that enables a clear presentation of data and efficient troubleshooting. On the back end, a multi-level error handling system was integrated that detects both API-side errors such as SOAP faults and database-related inconsistencies and provides precise solutions.

The developed "EUDR interface" ensures a secure, standardized and legally compliant transfer of all relevant data to the EUDR platform, thus fully meeting the compliance requirements of the EU Due Diligence Regulation.

1.3.5 / First Tests

Conducting the first tests of the EUDR interface proved to be more complex and time-consuming than expected, due to both technical and regulatory changes during the development phase. Originally, the testing effort had been estimated to be manageable, but adjustments to the EU-DDS API conformance tests led to additional challenges. In particular, the EU-DDS API error handling was updated during the course of the project, which meant that our system had to be adapted and retested. These changes mainly affected the processing of SOAP faults and the consistent reporting of errors to users.

In addition, legal changes complicated the development and testing process. Originally, it was planned that all companies would be required to submit Due Diligence Statements (DDS) from January 1, 2025. However, this requirement was adjusted multiple times and finally changed to remain optional in 2025. This meant that the system had to be adapted for a mixed case in which some companies already submit DDS while others do not. This new requirement necessitated additional programming work to seamlessly support the different scenarios, as well as extensive testing to ensure that all possible constellations could be processed correctly.

The combination of these factors led to a longer development and testing phase than originally planned. Nevertheless, the iterative adaptation and refinement of the interface allowed both the technical requirements and the amended legal requirements to be successfully integrated into the system.

1.3.6 / Further steps

The interface was set up and programmed so that each company in the Holz von Hier network can decide whether they want to use the Holz von Hier electronic control system to automatically submit due diligence declarations to the TRACES platform via the newly programmed interface. It did not make sense to have a mandatory link between the two, since the deliveries and transactions certified with HVH always meet the EUDR requirements, but not all goods flows that meet the EUDR requirements automatically meet the criteria of Holz von Hier.

Any questions? We are happy to help.

Contatcs:

LOW CARBON TIMBER ®
HOLZ VON HIER®

head office:
+ 49 (0) 9209 / 918 97 51
dr. Philipp Strohmeier and dr. Gabriele Bruckner

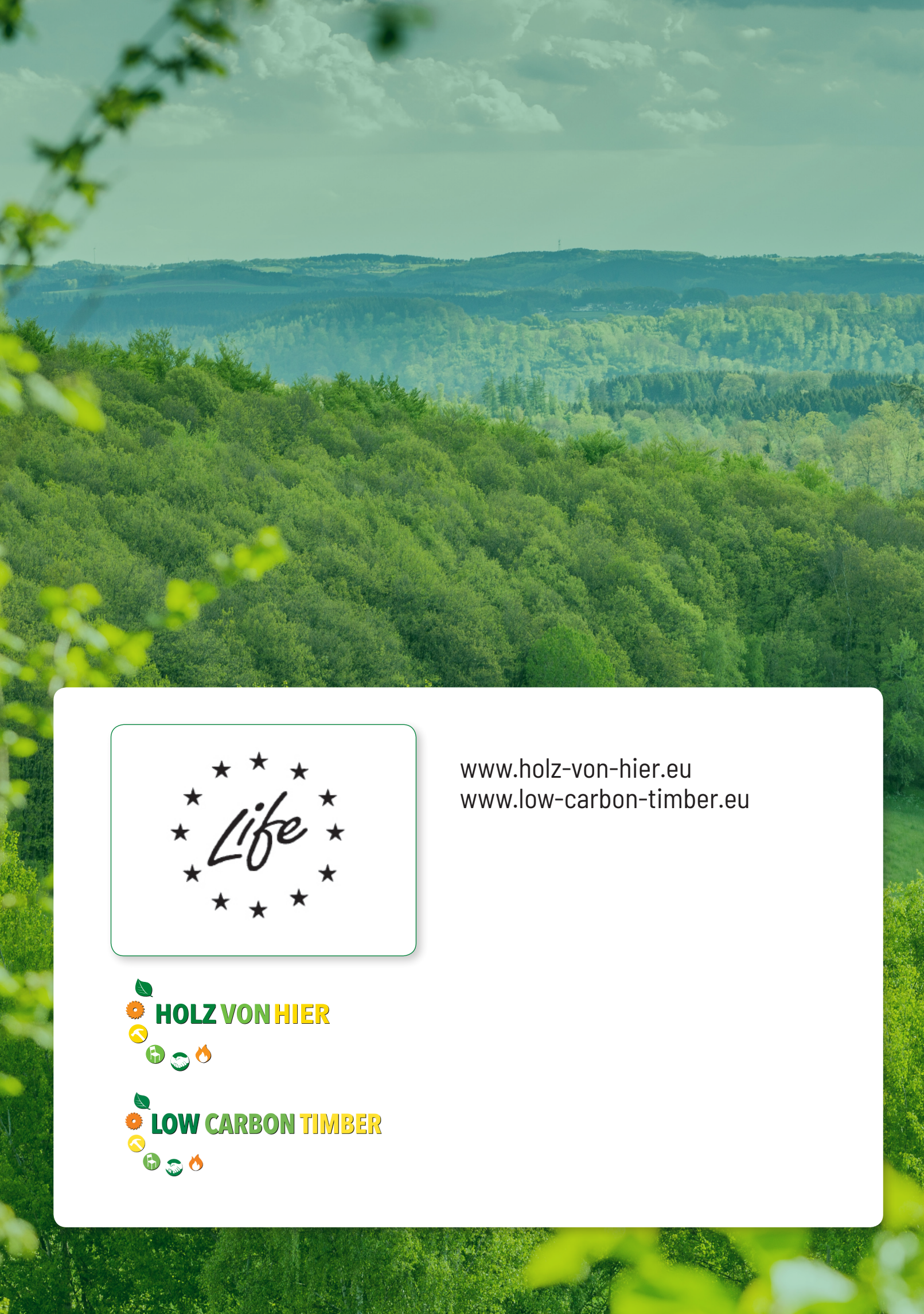
Service LCT/HVH Germany and Benelux:
+ 49 (0) 9209 / 918 97 51
Philipp Strohmeier, Gabriele Bruckner - Holz von Hier Germany

Service LCT/HVH Austria and Lichtenstein:
+ 43 (0) 664 / 3906478
Erich Reiner - Holz von Hjer Austria

Service LCT/HVH Italy:
+ 39 011 - 2257480
Luka Galeasso c/O ENVI Park Turino

Service LCT/HVH Slowenia:
Bernard Likar for enterprises fon.: +386 41 354 131 c/o Wood Cluster Slovenia-
Helena Cvenkel - for communities : fon.: +386 31302382 c/o BSC Kranj

LOW CARBON TIMBER



www.holz-von-hier.eu
www.low-carbon-timber.eu