Enabling Federated Interoperable Issue Management in a Building and Construction Sector

Oraskari J. [1], Schulz O. [1], Werbrouck J.[1,2], Beetz J.[1]
[1] Chair of Design Computation, RWTH Aachen, Germany, [2] Department of Architecture and Urban Planning, Ghent University, Belgium
jyrki.oraskari@dc.rwth-aachen.de, schulz@dc.rwth-aachen.de, jeroen.werbrouck@ugent.be

Abstract. A Common Data Environment (CDE) is an agreed-upon source of information on building-related projects to collect, manage, and exchange data between stakeholders. The approach in the AEC domain is to use buildingSMART's BIM Collaboration Format (BCF) as the digital issue communication part of CDEs. Contrasting with the federated nature of the AEC industry, CDEs are typically organised in a centralised fashion. This work proposes a potential transition of BCF into a distributed environment that serves as an example for further developments in the distribution of CDEs and CDE-independent data management. We show how a single source of truth over the project data and the advantages of the central approach can be realised in a distributed setup using a Solid architecture environment, enabling decentralised authentication and stakeholders' ability to control their data.

1. Introduction

In recent years, multiple international standards and specifications have been agreed upon to facilitate the information exchange in the Architecture, Engineering and Construction (AEC) industry. According to ISO 19650, a Common Data Environment (CDE) is defined as an agreed-upon source of information for a building-related project (ISO 19650-1, 2018). It should serve as a single source of truth and is therefore helping to structure the information and data exchange process in the course of a project between the different stakeholders.

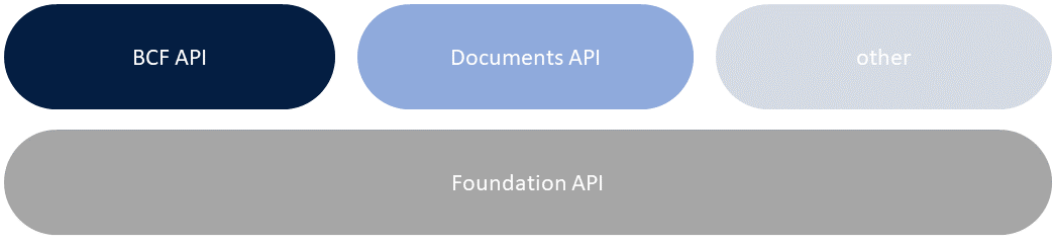

Figure 1: Abstraction of buildingSMART's OpenCDE APIs, originally presented by Yoram Kulbak and Pasi Paasiala in October 2019[1]. The BCF API is part of buildingSMARTs OpenCDE APIs.

The German standard DIN SPEC 91391-1 (2019) extends on the ideas of ISO 19650 and emphasises that a CDE for a building can change during the different phases of its life. Hence, there is a need for the different CDEs to communicate. The specification's authors suggest that standardised RESTful interfaces should be used to ensure the exchange of information containers between different CDEs at the BIM maturity stage 2.

A standardised approach for digital issue communication in AEC projects is the BIM Collaboration Format (BCF), which is commonly used in combination with CDEs (Preidel et al., 2018). BCF exists in two flavours: a file-based version and a centrally organised, server-based approach (van Berlo and Krijnen, 2014), allowing project managers and clients to study the entire body of issue descriptions via APIs to infer insights into the design team's work.

https://doi.org/10.7146/aul.455.c200

BuildingSMART International regards this server-based approach as part of the OpenCDE APIs[1] (Figure 1). Using BCF enables the user to communicate problems on a component-by-component basis, making BCF an integral part of the planning process. It supplements conventional e-mail communication about issues and can be seen as a central interface for communicating changes in the model. Hence, we focus on this format for this work.

Even though the standards mentioned above describe the exchange of information in detail, implementation remains scarce regarding container-based information exchange or the interconnection of CDEs. While streamlining information exchange is beneficial to the industry as a whole, it is not for commercial CDE providers. As in many sectors, the construction industry has followed the general trend toward Big Data and built data-driven ecosystems that centralise as much data as possible on their central servers. There are many challenges in this model. Data access is regulated via vendor-specific APIs, which often limit query parameters. Duplicated data in the silos can lead to a situation where the users do not have a single source of truth, making it harder to get insights into the project's status. Also, the legal aspects and the responsibilities of the data content, like the General Data Protection Regulation (GDPR) in the European Union, are harder to control if the data is not at the hands of their producers. The Solid initiative (Mansour et al., 2016) aims to change the overall course into a model where individual players control the data produced by them and about them. These players need a decentralised authentication mechanism included in the Solid specification for this to work.

This work proposes a potential transition of BCF into a distributed environment that can serve as an example for further developments in the distribution of CDEs and CDE-independent data management. We show how a single source of truth over the project data and the advantages of the central approach can be realised in a distributed environment, which enables decentralised authentication, lower risk of system failure, and stakeholders' ability to control their data.

The paper is structured as follows: the next section introduces the technologies and related works regarded for this paper. Section 3 describes the proposed way to express the federated structure of BCF data on the Solid platform. The paper concludes with a discussion on the proposed framework, including prospects on this topic.

2. Background

This section describes the background technologies on which the proposed framework will be based and related research projects. Although this paper bases heavily upon Semantic Web technologies, an elaborate discussion on these topics is outside the scope of this work. For a deeper understanding of technologies such as the Resource Description Framework (RDF) and the SPARQL Protocol, and RDF Query Language (SPARQL), the reader is referred to (Hendler et al., 2020).

2.1 BIM Collaboration Format and Common Data Environments

In the context of the ongoing establishment of open BIM processes, the need for a communication interface arose in order to be able to transmit information and issues within the models in a software-independent manner. Therefore, the BIM Collaboration Format (BCF) – a buildingSMART standard - was developed to provide a software- and vendor-neutral

[1] OpenCDE-API Documentation: https://github.com/buildingSMART/OpenCDE-API/tree/master/Documentation (Accessed 20.02.2022)

exchange format for model-based issue communication[2]. Three main parts - *Topics*, *Comments* and *Viewpoints* - together form the main structure behind BCF and are all connected to a *Project* concept:

1) The *Topic* carries general information about an issue. It uses properties like a current status, a definition of the type of *Topic*, the person who created it and who is responsible for fixing it.
2) *Viewpoints* are used to connect the format to BIM by providing a virtual camera located inside the model that looks at the scene that is part of a current discussion. They can also provide links to specific building elements by stating their GUID in a list.
3) A *Comment* concept provides the textual information and an author in a discussion. It is linked to the *Topic* and can reference a *Viewpoint* as well.

The BCF data can either be exchanged in a file-based format using BCF XML[3] or by using a server (van Berlo and Krijnen, 2014) via a REST API (called BCF API[4]) that returns its information in a JSON format. Even though the data can be serialised in two different ways, the overall concept behind both formats is the same, and the structures of BCF API and BCF XML differ only slightly. For example, BCF Servers - using the BCF API - often allow archiving and downloading of the *Project* and its issues as a file in the BCF XML format.

Since the communication in an open BIM process is usually an integral part of many workflows, it is often integrated directly into a Common Data Environment (CDE) (Preidel et al., 2018), which serves as a single source of truth throughout the planning and construction phase. Research regarding the decentralisation of these CDEs can also be observed. In Werbrouck et al. (2019), the authors suggest a decentralised CDE based on Solid principles, whereas Tao et al. (2021) describe a CDE distributed via a blockchain. The latter example also combines the principles of a CDE with BCF by distributing them as BCF XML over the blockchain.

2.2 bcfOWL

Since the current BCF serialisations lack the general contextual information and shared metamodels that RDF solutions have, the bcfOWL ontology (Schulz et al., 2021) was created to bring together the worlds of issue communication and Linked Data. In the proposed linked data model, BCF issues are stored as RDF triples and can be queried using SPARQL. Additionally, the different concepts for the *Topics* – defined in the BCF *Extensions* – can be enriched with further semantics. Therefore, the *Extensions* in bcfOWL can be used as a gateway to other ontologies in the context of Linked Building Data (LBD).

The ontology is not introducing new concepts to BCF and has semantic interoperability with BCF XML and BCF API. A converter can be created to serialise bcfOWL data into the standard BCF JSON and XML formats. Thus the ontology can serve as a shared foundation for both formats. By using SPARQL to query the data stored as bcfOWL, it is also possible to overcome the accessibility limitations of BCF caused by its hierarchical structure, as described in (Schulz and Beetz, 2021).

[2] BIM Collaboration Format (BCF) - An Introduction https://technical.buildingsmart.org/standards/bcf/ accessed 28.02.2022
[3] BCF XML: https://github.com/BuildingSMART/BCF-XML accessed 24.01.2022
[4] BCF API: https://github.com/buildingSMART/BCF-API accessed 24.01.2022

2.3 Authenticated Data Federation on the Web

Using the Linked Data Platform (LDP) specification[5], the Web of data can be accessed and managed using the read-write operations of the Hypertext Transfer Protocol (HTTP) standard version 1.1, the basis of data communication for the World Wide Web.

LDP incorporates the Linked Data principles (Bizer et al., 2009) of Tim Berners-Lee into a data container architecture. LDP relies on containerisation, where an ldp:Container refers to a specific, dereferenceable RDF graph listing its resources (ldp:contains). By dereferencing the container URL, it is easy to discover and, in turn, dereference its content (RDF or non-RDF) in a chain of HTTP requests. As container URIs are also RDF resources themselves, nesting containers is possible, resulting in a data organisation system that resembles file storage on a computer – yet now file paths are URLs.

While LDP on its own is well-suited for serving *open* data on the Web, it does not specify access-control mechanisms for protected datasets. Where a centralised data store often relies upon local storage of credentials, this is no viable option in a decentral environment. Since a client may combine hundreds of web resources to find what it needs, it is not feasible to maintain an account for these sources separately.

Established technologies like OpenID Connect (OIDC)[2] allow outsourcing this part of identity management to specialised identity providers (IDPs) that act as a service in the middle, e.g., Facebook, Google, or GitHub. The Solid initiative (Mansour et al., 2016; Sambra et al., 2016) eliminates the need for a third party: it provides the specifications to create an online identity based on a personal URL (a "WebID"[6]) on a domain chosen by the user. An office can thus become its own IDP and maintain its credentials for authenticating against decentralised construction management services (Werbrouck et al., 2019). A WebID is associated with a Personal Online Data storage ("Pod"): a data vault based on the LDP specification but now enabling fine-grained access control to containers and resources defined using the Web Access Control (WAC) [7] ontology. The resources that govern access control in Solid, ACL (Access Control List) resources use the WAC ontology to link specific access rights to specific WebIDs, agent groups *(vcard:Group*) or (un)authenticated agents (*acl:Agent* or *acl:AuthenticatedAgent*). This way, it can be easily verified by a Solid Pod provider whether an actor can interact with a resource in a specific way: read, append, write or control, i.e., modifying the ACL document itself.

Apart from the ACL resources that govern access rights, the Solid specifications define '.meta' resources: RDF resources that contain metadata statements related to an ldp:Container instance. A .meta resource is served upon dereferencing a container URL. The details of .meta resources are yet to be agreed upon within the Solid community, but by default, they include the LDP containment triples and modification dates. Attaching a custom, persistent .meta resource to a Solid container with domain-specific metadata is possible.

2.4 LBDserver

A related initiative that uses Solid to store AEC data in a decentral way is the ongoing LBDserver project (Werbrouck et al., 2021). This project proposes data structures to discover project resources, metadata storage and cross-document linking of heterogeneous datasets using

[5] Linked Data Platform: https://www.w3.org/TR/ldp/ accessed 28.01.2022
[6] WebID 1.0: https://www.w3.org/2005/Incubator/webid/spec/identity/ accessed 31.01.2022
[7] Web Access Control, https://solid.github.io/web-access-control-spec/, accessed 31.01.2022

the LBDserver vocabulary[8]. Throughout this paper, we will re-use certain patterns proposed in the LBDserver ecosystem (Section 3). However, where the LBDserver proposes a very generic way for data organisation, the BCF specification has a distinct way of structuring BCF-related datasets. As this is standardised within the AEC industry, we will maintain this way of data organisation in this work.

3. Federated BCF projects

In this section, we sketch the outline for the setup of federated BCF projects. Therefore, we combine domain-agnostic Web specifications from the Solid ecosystem with domain-specific concepts proposed in the bcfOWL and LBDserver initiatives. Section 3.1 describes an organisational structure for discovering and managing federated BCF data. The project becomes a federated graph in such a setup: the union of contributions stakeholders make on their own "office server", shared using WebIDs. Similarities with the existing BCF API and BCF XML will be drawn where relevant. Based on Solid's existing Web Access Control specifications, an access-control layer is devised upon this organisational structure.

3.1 Project Discovery

To make federated BCF projects easily discoverable, we base upon the aggregation structures proposed in the LBDserver. This means that an office can maintain its projects in a root *ldp:Container* on its Pod, i.e., the Project Repository. The URL of this registry is referenced in the office's WebID (*lbds:hasProjectRegistry).* A project registry has sub-containers for each

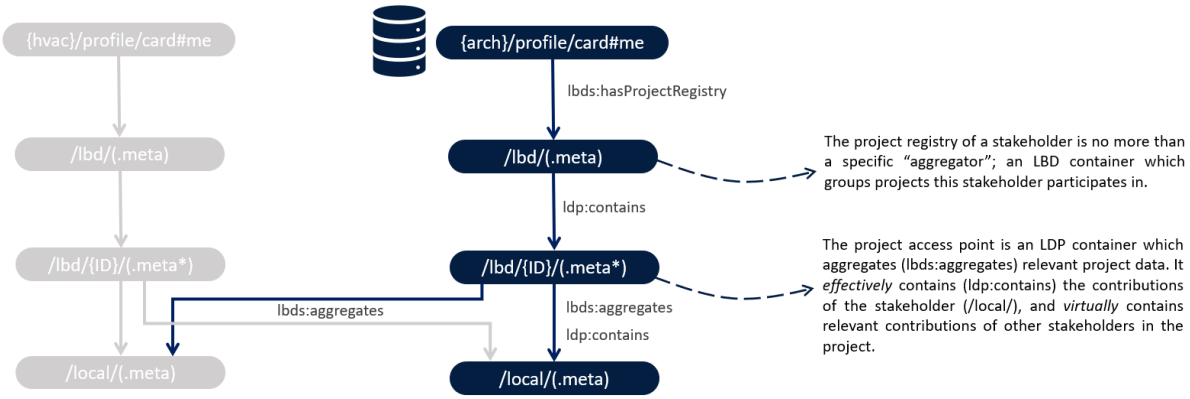

Figure 2: LBDserver patterns for project discovery. The project access point allows finding contributions in the federated project network.

project the Pod owner participates in ('project access points'). These sub-containers, in turn, have pointers to the contributions ('partial projects') of each stakeholder, including the contribution of the owner of the Pod, which we identify by a </local/> sub-container (Werbrouck et al., 2022). The first type is "virtually contained", and the second type is effectively hosted by this container on the Solid Pod. Thus, by dereferencing the project access point, a client discovers a list of federated partial projects.

[8] The LBDserver vocabulary, https://w3id.org/lbdserver#, accessed 08/02/2022

This paper focuses on arranging the BCF data (i.e. *Projects*, *Topics*, *Viewpoints* and *Comments*) inside a Pod, leaving the storage of auxiliary resources (e.g. PDF documents, IFC models) out of scope. In this regard, we suggest a tree-like structure that mimics the routes in the BCF API

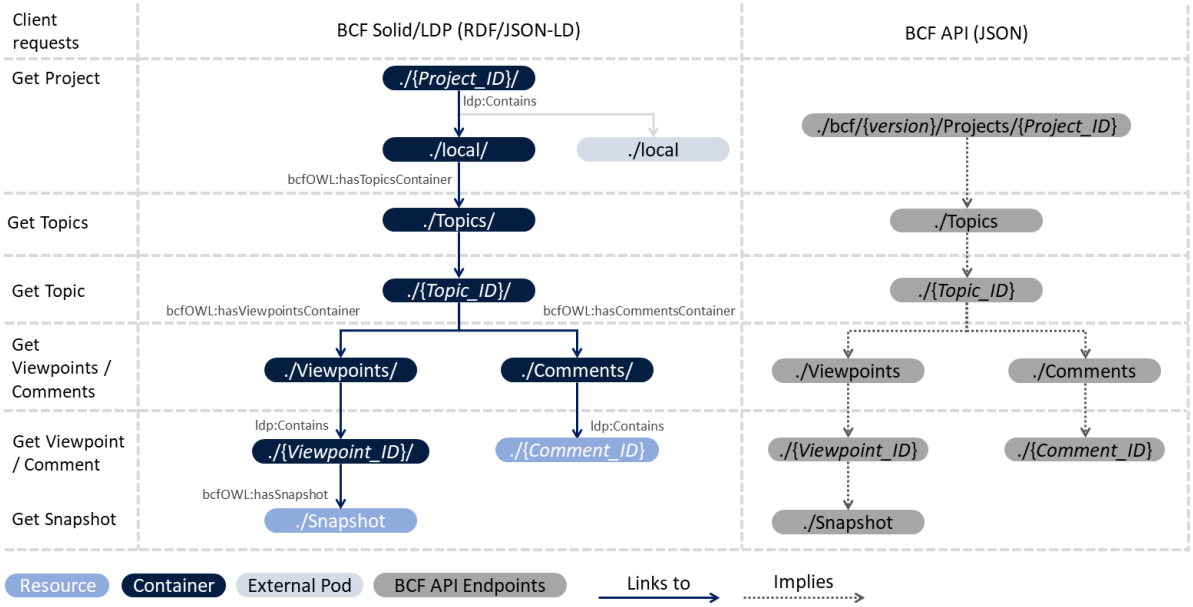

Figure 3: Comparison between the structure of the container-based BCF Solid approach and the BCF API (buildingSMART). The structure of the BCF API is not discoverable by the client without previous knowledge of the standard (it is implied). The BCF Solid approach implements a machine-readable discovery pattern (RDF graph) by linking to its sub-containers.

and the folder structure in BCF XML (Figure 3), in contrast to the flattened approach in the LBDServer.

The metadata resource corresponding with the </local/> folder (i.e. at the top level) contains general information about the *Project*, such as its name, what BCF *Extensions* it contains and how they are defined. Furthermore, it links to one or many sub-containers that contain information about the *Topics* belonging to the *Project*, using dedicated sub-properties of "*ldp:contains*". These properties are described in detail in the following section.

3.2 Project organisation

Sub-containers containing BCF *Topics* are identified with the RDF predicate "*bcfOWL:hasTopicsContainer*". Each *Topic* is itself located in a sub-container in this container, described with a metafile that contains the *Topics* information in bcfOWL and links to its *Viewpoints* and *Comments* by using the predicates "*bcfOWL:hasViewpointsContainer*" and "*bcfOWL:hasCommentsContainer*". These sub-containers correspond in their structure to the *Topics* route defined in the BCF API by including *Viewpoints* and *Comments*. Each *Viewpoint* is a container in itself. The *Comment* is represented as a resource (Figure 3). The *Viewpoint*, in turn, contains more concepts, such as a *Snapshot* (e.g. .png or .jpeg) or a *Perspective Camera*.

If we take a closer look at this hierarchy (Figure 3), we can see that there are many similarities to the server routes of the BCF API. Whereas in BCF XML, each issue is located in its folder (identified by the GUID of the *Topic*). The *Topic* is summarised together with the *Comments* in a Markup file. The BCF API provides all this information as hierarchically organised REST API URI patterns that are accessed sequentially. The BCI APIs sequence is mainly preserved in our proposed structure and allows finding data using standardised endpoints. Apart from the fact that every partial project acts as a "partial BCF server" on its own, which means they need

to be queried using multiple HTTP requests, a client will not experience a difference between a Solid-based *Topics* container and a centralised one. A single Pod can be used as a complete BCF server without referencing external partial projects. However, the same infrastructure can be used to federate the project's information.

However, some benefits arise with the RDF-based organisational approach compared with the mere implementation of standardised API routes. Although in an LDP (Solid) environment, similar endpoint patterns to the BCF API are used, these endpoints are also semantically described using the metadata files. Instead of just receiving BCF JSON responses from the server, the server describes what each container entails and what its metafiles depict. The information is easily discoverable in these containers on the Pod. Because resources are stored as files but served in a REST API, it combines the file-based BCF XML and the service-oriented BCF API approaches.

Furthermore, the links to the sub-containers do not have to be restricted to local resources and can point to any number of other Pods from other stakeholders. Thus, a distributed communication of issues is achieved, in which each participant can store remarks and additions in their Pod.

Lastly, this approach allows the dynamic discovery of data. Although the proposed tree structure mimics the BCF API and the BCF XML structure, it is not the only possible configuration. The property-based discovery of containers and sub-containers allows a client to discover how a project is organised. However, this resource could have been stored in a completely different location. An external service may then present this data "as if" it is compliant with a specific standard such as BCF.

3.3 Access Control and Groups

In a decentral project, each office may maintain access control to the resources they contribute. The WAC ontology supports *acl:AgentGroup*-s, which point to a *vcard:Group* instance, referencing its members (v*card:hasMember*) via their WebID. For each project, an office can publish one or more groups containing the employees' WebIDs (e.g., 'localEmployees.ttl'). The ACL that governs the resources in the local project folder can then grant these groups specific access rights. For instance, the responsible project manager in the office gets *acl:Control* rights and the *acl:Read* and *acl:Write* rights of the other employees working on this project. Agent groups defined by other project stakeholders (i.e., hosted in their project container) are granted only acl:Read permissions. Although the above describes a project-specific approach, this works similarly for resource-specific access rights.

3.4 Project interaction

This paper does not yet tackle the challenges of a complete workflow, where people create, comment, and update new and existing Issues. When we take the update of an issue as an example, the question must be asked, where this update will be stored in the distributed system. A possible option is that the person who creates the update is storing the new issue - or just the updated content – on their Pod and notifies the original Pod of the issue of its existence. Thereby, when the issue gets queried, it returns a reference to its new version. This option allows tracing the complete history of the *Topic* without ever really deleting or changing existing data. However, it also creates much redundancy that could, in the long run, influence the performance of the queries. Another option that is more in line with the current implementation of the BCF API is to implement a logging system that keeps track of the changes by the users. This topic is further discussed in Oraskari et al. (2022).

4. Proof of Concept

A proof of concept was created to demonstrate the feasibility of the proposed framework for federated management of BCF data. This includes using the Solid Community Server[9], an open-source implementation of the Solid specifications. The data used in this demonstration is based on the BIM models of the DC chair at RWTH Aachen University[10].

To emulate the federated environment, three Pods were set up, containing a total of six issues:

- The Pod owned by the project architect office contains four topics
- The Pod owned by the HVAC engineer contains one topic
- The Pod owned by the structural engineer contains one topic

In this demonstrative scenario, Oliver, working at the engineering office, wants an overview of the current issues registered for the DC chair project. He is registered at the Architect Office's Pod in the list of employees assigned to this project (Listing 1).

Listing 1: the employee group (http://pod.myoffice.org/Projects/8b71315b-7db2-4b35-a1e9-fcddaf8556f5/groups#committed)

```
<#committed> a vcard:Group ;
    dc:created "2022-02-22T22:22:22Z"^^xsd:dateTime ;
    dc:modified "2022-02-22T22:22:22Z"^^xsd:dateTime ;
    vcard:hasMember <http://localhost:3000/oliver/profile/card#me> ,
            <https://localhost:3000/jeroen/profile/card#me> .
```

Every stakeholder's </local/> partial projects contain an .acl file that references this group and assigns to acl:Read rights to its members (Listing 2). Of course, editing rights (acl:Write, acl:Append) can be granted in the partial project provided by the architect's office itself.

Listing 2: access rights for employee group in Listing 1

```
<#us>
    a acl:Authorization;
     acl:agentGroup  <http://localhost:3000/office/Projects/8b71315b-7db2-4b35-a1e9-
fcddaf8556f5/groups#commited> ;
    acl:accessTo <./>;
    acl:default <./>;
    acl:mode acl:Read, acl:Write, acl:Append .
```

First, the available partial projects need to be discovered. The SPARQL query covers this:

```
SELECT ?partial WHERE {<> lbds:aggregates ?partial}
```

As indicated in Figure 3, the organisation of partial projects corresponds with the BCF API specification data patterns. Because these are standardised routes, further discovery is not necessary, and the Web client can send the following (authenticated) requests to each of the partial projects:

```
GET {partial project}/topics
```

This will yield an LDP container with pointers to the contained *Topics*, which can now be easily retrieved and presented in a GUI. This series of HTTP requests is not identical to those required to access the BCF API. However, they can be easily implemented under the hood by a BCF server to expose federated information to conform to the standard. In that case, the server acts as a middleware to regulate access to information on the stakeholder Pods.

[9] Solid Community Server, https://github.com/solid/community-server, accessed 28.01.2022
[10] Demo dataset: https://github.com/Design-Computation-RWTH/EG-ICE_2022_BCFdemo accessed 28.02.2022

5. Discussion and Conclusion

The translation of BCF into a Solid environment introduces its main concepts in a federated setup, using the Solid specifications and the Linked Data Platform. This way, issue communication can be spread over multiple Pods that belong to different stakeholders. *Project* stakeholders are no longer solely identified by their e-mail addresses, as implied by the BCF user concept. Instead, the use of semantically rich WebIDs for offices and employees allows the dynamic creation of user groups (e.g. defined by role or participation) and the re-use of credentials in multiple federated *Projects*. A project team from an architectural stakeholder can define its members in their Pod and manage who has access to the *Project* data and who does not. The resulting group (vcard:Group) is then linked to BCF *Project* data to control access. Hence, it becomes possible to define more granular access rights for the different roles on the *Project* pod. This corresponds to the structures in the federated construction industry.

The decentral authentication mechanism, as defined in the Solid specifications, does not require the server to store the login credentials for every user. The service is just responsible for verifying if the access rights associated with a given WebID allow a user to interact with a specific resource in a specific way. The need for the user to create an account for every service is thereby removed by using WebIDs.

Future developments in this area include investigating the guaranteed availability of the federated data to prevent the loss of project information (either accidentally or intentionally). Furthermore, a notification system between Pods in the network will enable automated synchronisation (e.g. when someone updates a Topic or creates a new Comment). In this paper, we proposed a tree-like structure that combines the traits of BCF XML and the BCF API. The URL routes of the BCF API thereby serve as a means to access the individual containers. Information in these containers is stored in a .meta file, and the different containers holding the BCF data are linked via bcfOWLs properties. We have shown that BCF projects information can be queried using a one-stop access point implemented using distributed Solid architecture and offered as a single source of truth. In the platform, authentication can be self-hosted, and the stakeholders can manage their data using a tree-like structure that mimics the hierarchical pattern of the BCF API endpoints. It was also shown that the data authorisation enforcement can be set using the container architecture of the Solid specification.

Since BCF is a member of the OpenCDE API family, we hope this research can serve as a template for further research on aligning AEC standards with future-proof concepts for federation on the Web.

Acknowledgements

This research is funded by the Research Foundation Flanders (FWO) as a Strategic Basic Research grant (grant no. 1S99020N) and by the EU through the H2020 project BIM4REN grant (grant no. 820773).

References

Bizer, C., Heath, T., Berners-Lee, T., 2009. Linked Data - The Story So Far. Int. J. Semantic Web Inf. Syst. 5, 1–22. https://doi.org/10.4018/jswis.2009081901

DIN SPEC 91391-1 Common Data Environments (CDE) for BIM projects - Function sets and open data exchange between platforms of different vendors - Part 1: Components and function sets of a CDE; with digital attachment, 2019.

Hendler, J., Gandon, F., Allemang, D., 2020. Semantic Web for the Working Ontologist: Effective Modeling for Linked Data, RDFS, and OWL. Morgan & Claypool.

ISO 19650-1 Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) - Information management using building information modelling - Part 1: Concepts and principles, 2018.

Mansour, E., Sambra, A.V., Hawke, S., Zereba, M., Capadisli, S., Ghanem, A., Aboulnaga, A., Berners-Lee, T., 2016. A Demonstration of the Solid Platform for Social Web Applications, in: Proceedings of the 25th International Conference Companion on World Wide Web, WWW '16 Companion. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, pp. 223–226. https://doi.org/10.1145/2872518.2890529

Oraskari, J., Schulz, O., Beetz, J., 2022. (in press) Towards describing version history of BCF data in the Semantic Web, in: Proceedings of the 10th Linked Data in Architecture and Construction Workshop.

Preidel, C., Borrmann, A., Mattern, H., König, M., Schapke, S.-E., 2018. Common Data Environment, in: Borrmann, A., König, M., Koch, C., Beetz, J. (Eds.), Building Information Modeling: Technology Foundations and Industry Practice. Springer International Publishing, Cham, pp. 279–291. https://doi.org/10.1007/978-3-319-92862-3_15

Sambra, A., Mansour, E., Hawke, S., Zereba, M., Greco, N., Ghanem, A., Zagidulin, D., Aboulnaga, A., Berners-Lee, T., 2016. Solid : A Platform for Decentralized Social Applications Based on Linked Data.

Schulz, O., Beetz, J., 2021. Image-documentation of existing buildings using a server-based BIM Collaboration Format workflow., in: EG-ICE 2021 Workshop on Intelligent Computing in Engineering. Presented at the EG-ICE, Berlin.

Schulz, O., Oraskari, J., Beetz, J., 2021. bcfOWL: A BIM collaboration ontology, in: Proceedings of the 9th Linked Data in Architecture and Construction Workshop. Presented at the LDAC2021, Luxembourg.

Tao, X., Das, M., Liu, Y., Cheng, J.C.P., 2021. Distributed common data environment using blockchain and Interplanetary File System for secure BIM-based collaborative design. Autom. Constr. 130, 103851. https://doi.org/10.1016/j.autcon.2021.103851

van Berlo, L., Krijnen, T., 2014. Using the BIM Collaboration Format in a Server Based Workflow. Procedia Environ. Sci., 12th International Conference on Design and Decision Support Systems in Architecture and Urban Planning, DDSS 2014 22, 325–332. https://doi.org/10.1016/j.proenv.2014.11.031

Werbrouck, J., Pauwels, P., Beetz, J., Mannens, E., 2021. Data Patterns for the Organisation of Federated Linked Building Data, in: Proceedings of the 9th Linked Data in Architecture and Construction Workshop. Presented at the LDAC2021, Luxembourg.

Werbrouck, J., Pauwels, P., Beetz, J., van Berlo, L., 2019. Towards a Decentralised Common Data Environment using Linked Building Data and the Solid Ecosystem.

Werbrouck, J., Pauwels, P., Beetz, J., Mannens, E., 2022. LBDserver - a Federated Ecosystem for Heterogeneous Linked Building Data. (Under review).