CV 3.0 Content Access Service —Architecture and Proof of Concept Implementation Specification

Pascal Mainini

Berner Fachhochschule

Technik und Informatik

Quellgasse 21, CH-2501 Biel/Bienne

pascal.mainini@bfh.ch

Document ID: http://cv3.bfh.ch/architecture-implementation.html

1 License

2 Created 2013-04-12

Abstract. CV3.0 is a research project of the Bern University of Applied Sciences (BUAS) to explore a new way of presenting educational achievements and scientific activities in electronical form and online. It enhances the data of a traditional curriculum vitae with additional information and introduces a platform for accessing such a CV3.0 curriculum vitae. This document gives an overview of the platform architecture and specifies the proof of concept (PoC) implementation created at BUAS.

Keywords: Linked Data, Curriculum Vitae, WebID

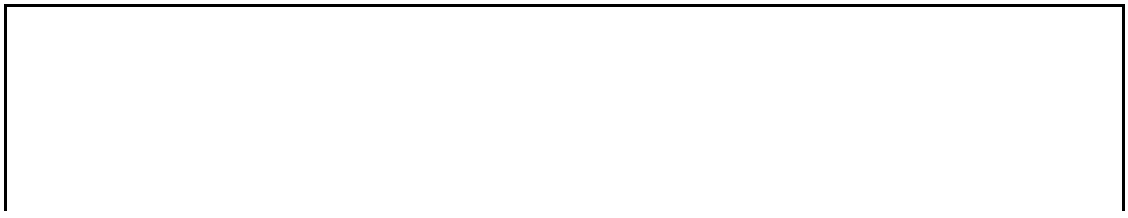
Part I. - Architecture overview

3 Introduction

In a CV3.0-environment, multiple systems and components interact to build and provide an online curriculum vitae. This first part describes these systems and their interaction in an abstract manner; a specific implementation is given in the second part which describes the proof of concept implementation at BUAS [1].

4 System Overview

Figure 1 depicts the CV3.0-environment and the involved systems and actors.



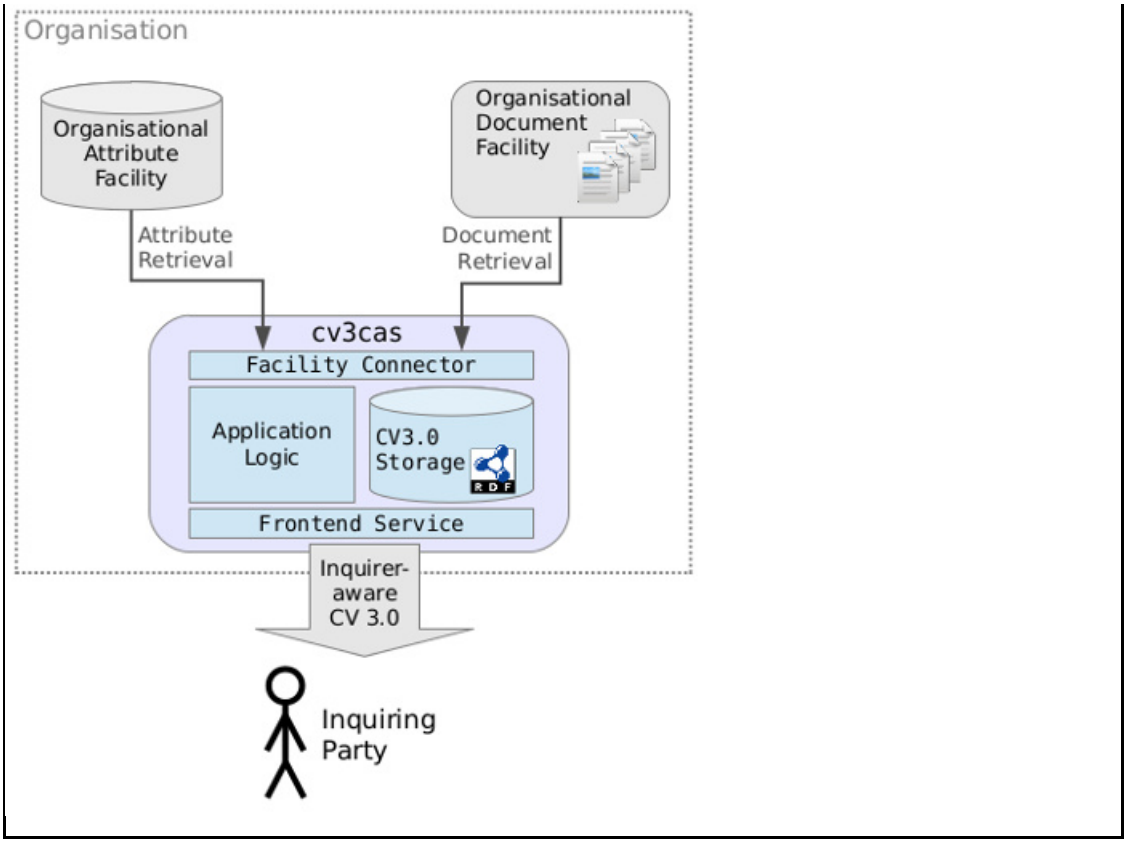


Figure 1: Overview of cv3cas and surrounding systems

In the following sections, all parts will be described.

5 Inquiring Party

The inquiring party defines the actor who requests CV-data from the CV3.0-environment, it can be either human or an application/service. Different classes of inquiring parties stand for different levels of trust and differ in restrictions while accessing a CV3.0. The person which is associated with the CV itself by definition has more rights than an employment agency or even an anonymous robot crawling the web. The proposed system must consider these different kinds of inquiring parties and act accordingly.

6 Inquirer-aware CV3.0

The full content of a CV3.0, referred to as *CV3.0* or *CV3.0-content* throughout this document, contains the additional CV3.0-data as well as all the data coming from the organisational facilities depending on the access level of the specific inquiring party requesting it.

7 CV3.0 Content Access Service (cv3cas)

At the front, the *CV3.0 Content Access Service (cv3cas)* provides access to CV3.0-content for inquiring parties. It serves as a proxy for accessing data from the attribute- and document-facilities of an organisation (see section Facilities), as well as for accessing additional, CV3.0-specific content.

This functionality is implemented in 4 core components which are described in the following.

7.1 Components

7.1.1 Front-end Service.

The front-end service provides access for inquiring parties to CV3.0-content. Content is accessed on a per-CV-basis: The common denominator is a *CV* which is related to a single person. The CV bundles all the data coming from the organisational facilities as well as the additional CV3.0-content. Depending on the trust-level of the inquiring party, more or less content may be retrieved, see section Trust.

Depending on the inquiring party (human, owner, agent, ...), specific representations of the content are returned.

7.1.2 Application Logic.

The core application logic represents the glue between the different components of cv3cas and interacts with them.

7.1.3 CV3.0-storage.

CV3.0 introduces additional attributes and metadata on top of the data received from the organisational facilities. The whole set of data including everything related to a specific CV retrieved from any facility as well as the CV3.0-content itself is stored internally in the CV3.0-storage in RDF-format [2].

7.1.4 Facility Connector.

The facility connector is responsible for the connection to the organisational facilities for documents and attributes. It provides a plugable interface to access various kinds of backend-systems, like for instance relational database systems or fileshares.

7.2 Other aspects

Besides the technical implementation details of cv3cas, some other important aspects surrounding security and data protection must be carefully considered in the design and the implementation.

7.2.1 Security, Trust and Privacy.

Dealing with potentially sensitive information, special attention must be given to issues of security, trust and privacy.

On a first level, communication security and confidentiality must be ensured. This is achieved by using well-defined and widely accepted standards such as TLS [3]. All communication with inquiring parties on the front side of cv3cas must be secured according to best current practises. In communication with organisational facilities, the securest method offered by those is used where applicable.

For privacy in terms of personal data protection, different access levels depending on the inquiring party must be implemented. Consider the following examples for clarification on this:

- An anonymous inquiring party can only enumerate the available CVs, however, not request their content
- Retrieval of basic CV-data like academic degrees for inquiring parties with basic trust (e.g. employment agencies)
- Retrieval of full CV-data for inquiring parties specifically allowed by the owner of the CV and by the owner itself

7.2.2 Electronic Signature.

The content of a CV is used by the inquiring party in comparing persons, for instance when applying for an employment. Thus, the contents of a CV must be correct and this correctness has to be proven by the issuing organisation (or all involved organisations in a distributed case).

Digital signatures are now standard in assuring such proof and are also to be used within CV 3.0. Two cases have to be differentiated:

1. Signatures issued by cv3cas itself
2. Signatures on data received from the organisational facilities described in the next section

In the first case, cv3cas can issue the signature itself and guarantee it's correctness; in the second case, the signature from an organisational facility will be relayed as-is and a statement can be made that the signature has been obtained at a certain point in time.

8 Organisational Facilities

Not all the data in a CV3.0 is generated by the cv3cas itself. Additional data is coming from surrounding systems within the organisation. Two types of data can be distinguished:

- Data in the form of *documents*, examples would be a master thesis or other papers written by the person, photos etc.
- Data in the form of *attributes*, like a person's email-address or birth date.

Systems, which are queried by cv3cas for such data, are called *organisational facilities*, as they belong to and are run by the organisation which hosts the cv3cas. It is possible for an organisation to have more than one facility of every kind. The two types of organisational facilities will now be described in more depth.

8.1 Organisational Document Facility (ODF)

The *organisational document facility*, in short *ODF*, is a facility which stores documents (or more exactly: files), which are related to the CV. This could include photos or other material which accompanies the CV of the person, however, the primary use for this facility is the storage and retrieval of certificates and other forms of confirmations for achievements at an organisation.

An organisational document facility can range from a traditional fileshare (Windows- or WebDAV for instance) up to a full-blown document management system (DMS [4]).

8.2 Organisational Attribute Facility (OAF)

An *organisational attribute facility*, in short *OAF*, is a facility which stores and provides attributes about a person within an organisation. Typically, such attributes are used in the domain of identity management [5] and federated identity management [6].

Examples for organisational attribute facilities would be an LDAP- or any other directory service [7], an identity federation, like SWITCHaai [8] or other internal systems, like for instance systems used by human resources departments.

Part II. - Proof of Concept Implementation

While the first part of this document described the involved systems, components, actors and their relation to each other, the second part now focuses on the proof of concept implementation built during the CV3.0 research project at BUAS. It will describe the technologies used for building and interconnecting the individual parts, as well as the data transmitted between them and included in a final, PoC CV3.0.

Note: the technical overview given in this part is meant explicitly as an overview of the technologies and methods used for the PoC-implementation, as well as - for certain specific areas - as a research summary. It is not an in-depth technical specification at the implementation level. Further documentation will be included with the built PoC.

9 Organisational Facilities involved in PoC

In the proof of concept, a first (as the name implies) PoC-grade implementation of a cv3cas-service will be implemented and connected to organisational facilities at the BUAS.

9.1 SQL-Database as OAF

At BUAS, currently two possibly interesting attribute facilities have been identified:

- IS-Academia (IS-A), a student directory containing all administrative data related to a student

- myIDP, a claim-assertion-service for the SuisseID [9], developed at BUAS

As both attribute facilities rely on SQL databases as backend for storing the attributes, a facility connector for SQL-databases will be built during the PoC.

9.2 Simple ODF in a git-RCS

In a preliminary analysis, the need for a document management system capable of issuing digitally signed documents has been identified. Further research amongst existing DMS-solutions showed that these either don't offer the required signature feature in a form usable by the project or are too complex to install, configure and maintain for the PoC.

It has thus been decided to implement a simplified DMS-system using technology well known to the research group and use that as an *exemplary* DMS.

For versioning and integrity control in the simulated DMS, the widely deployed revision control system *git* [10] will be used. git works as key-value-store where any kind of data which is added to it can be referenced by its *SHA-1*-hash [11]. As example, figure 2 depicts a revision of the PDF-version of this paper in its git-repository [12].

```
commit 03f340691bb90870f19fda45678fd824c93ed53f
Author: Pascal Mainini << pascal.mainini@bfh.ch >>
Date:   Fri Apr 12 16:45:21 2013 +0200

   initially added
```

Figure 2: git-commit information, SHA-1-hash highlighted

An additional layer of trust could be added by using an X.509 trusted timestamping-service for each commit to the ODF, similar to this approach [13]. However, it has been decided that digital signature of the documents in the facility is out-of-scope and should be done by the issuer of a document and within the document itself.

10 cv3cas PoC Implementation

The PoC of cv3cas will be implemented as a server-side application in JavaScript using the node.js-framework [14]. node.js is a modern, asynchronous, event-driven framework for implementing primarily serverside applications in JavaScript. The researchgroup gained knowledge in using node.js already in other projects and has made it to its current standard-platform for development.

10.1 Facility connectors

As part of the PoC, a plugin-architecture for facility connectors will be implemented. This architecture permits the use of interchangeable plugins to connect to various types of attribute/document facilities. Plugins have to provide an abstract interface to the respective type of facility and are always implemented in a read-only manner. Two types of operations on facilities have to be defined:

Enumerate Retrieve a list of attributes/documents belonging to a CV

Retrieve Obtain a specific attribute/document belonging to a CV

In both cases, identification of the CV is done using the unique identifier defined in section Unique Identifier.

10.2 CV3.0-Storage

As back-end platform for the CV3.0-storage-component, Fuseki [15] of Apache's Jena-project has been choosen. Fuseki is an RDF-triplestore [16] which has already been deployed in other projects of the research group at BUAS and which is by far sufficient for the needs of the PoC.

10.3 RESTful Front-end Service

The front-end service will be implemented as a *REST*-based [17] software architecture. REST is built around the

HTTP-Protocol [18] and is a client-/server-based architecture using the concept of resources and representation of these in "documents". State and modification of the resources is achieved using HTTP-methods. The same endpoint can be used by machines as well as humans.

10.3.1 Endpoint.

A CV can be accessed over the REST-endpoint using the relative URI `/cv/<identifier>` where identifier is the unique identifier of a CV. See section Unique Identifier for details of the identifier.

The REST-endpoint of the cv3cas supports *content negotiation* [19], thus returning a suitable representation depending on what has been requested by the HTTP-agent. Table 1 lists the supported content-types.

Content-Type	**Description**
`text/html`	Human-readable CV-representation
`text/turtle`	Human- and machine-readable RDF representation of CV in *turtle* [20]
`application/rdf+xml`	Machine-readable RDF-representation of CV in RDF+XML [21]
`application/json` (optional)	Either JSON [22] or JSON-LD [23] representation of CV

Table 1: Content-types supported by the REST-endpoint

It is planned to build the HTML-interface for human interaction using *uduvudu* [24], a framework for building user interfaces for RDF-data in RDF, currently being developed at BUAS. If the implementation of uduvudu is not mature enough by the time it is needed within the CV3.0-project, a simplistic and hardcoded view on the data will be implemented.

The JSON-content-type is optional and will only be implemented if the other parts could have been completed in time. JSON can nowadays be considered as industry-standard for interaction between applications on the web and would thus facilitate the exchange with other applications, however, for demonstration of the completeness of the PoC it is not necessarily required.

10.3.2 Unique Identifier.

As unique identifier for CVs, the *SwissEduPersonUniqueID* [25] has been chosen as it is guaranteed to be unique within the whole educational landscape of Switzerland. It also should be, according to its specification, non-transparent, meaning that it should not be possible to identify a person only through their SwissEduPersonUniqueID.

Another possibility for the identifier would have been the BUAS-specific shortcut of a person which would also be unique BUAS-wide. Eventually, the PoC will offer support for accessing a CV in addition using the BUAS-shortcut.

10.3.3 Authentication.

As described in section 5.2.1, access to a CV3.0 has to be limited depending on the inquiring party. The inquiring party thus has to be authenticated against the front-end service. In the PoC, this authentication will be performed using WebID [26] as well as a BUAS-specific WebID-"IDP" [27]. Table 2 lists the different roles, how they are identified and a short description.

Role	**Identified by**	**Access to**
CV3.0 Owner	BUAS WebID, matching unique ID [28]	Full CV3.0 content
BUAS User	BUAS WebID, different unique ID	Information commonly visible within BUAS
Foreign Party	Any WebID, possibly no unique ID	Depending on user consent [29]
Anonymous	Unauthenticated	Only minimal content

Table 2: Roles, identification and permissions

A BUAS-specific WebID-"IDP" will also be implemented separately. It provides a BUAS-specific WebID-profile containing the unique ID of the user.

10.3.4 Authorisation.

After successful authentication as described in the section before, authorisation to access various levels of CV3.0-content has to be determined based on the inquiring party, respectively it's agent. In the semantic web environment, standards for authorisation and access control are currently evolving, the following could have been identified as the most promising / mature ones:

- WebACL [30] developed by W3C, strong connection to WebID
- Shi3ld [31] developed by INRIA, more powerful / complex

For the PoC implentation, WebACL has been chosen due to its greater simplicity and its tight integration with WebID used for authentication. Also, the granularity provided by WebACL seems fine enough for the needs of the project.

It has to be noted, however, that there could not have been found any implementations in JavaScript, thus increased development time must be considered. The same, however, applies to Shi3ld as well.

10.3.5 User Consent.

As multiple and different kinds of inquiring parties access a CV, different amount and detail of CV-content is delivered. The owner of a CV must be able to decide to which inquiring party which level of access is given. Multiple possibilities of implementing the user-consent have been identified:

- Automatic consent: a set of inquiring parties is defined and the user can choose the level of access for each one independently
- Per-request consent: upon each request of an inquiring party, the user can decide to grant the request or deny it
- Fixed definitions of inquiring parties and user consent (*PoC only*)

In the first case, the inquiring parties are known to the CV3.0-environment and the user can decide to allow one or multiple of them individually. This approach can be compared to the "app-model" of Facebook, Twitter and the like.

In the second case, every request for content which is not available by default to an inquiring party requires user interaction. This is a semi-automatic, asynchronous approach.

For the PoC-implementation, in order to keep the development effort reasonable, a third option has been chosen: A predefined set of inquiring parties is defined within the PoC and user-consent is matched statically against this set using a lookup-table.

10.4 CV RDF Data Format

As part of his bachelor thesis, Joel Cabral develops the RDF-ontology to represent the contents of a CV3.0 in RDF. This ontology will be used as the base model for the PoC-implementation of CV3.0, the REST-endpoint will return RDF-data according to the specification given by it.

10.4.1 Digital Signature.

In a CV3.0, digital signatures take place at two locations:

- In the *Organisational Document Facility*: Storage of digitally signed documents
- In the *CV3.0-Content* itself: Digitally signed RDF data

While the first case is out of scope of the PoC, the implementation has to handle the second case.

Digitally signing RDF brings up various problems (like for instance normalisation and serialisation), which have already been addressed theoretically as well as with actual implementations. For the theoretical background, refer to Carroll [32].

Payswarm [33] has implemented signing, verification as well as encryption/decryption for JSON-LD as part of their client-API [34]. The PoC will base the digital signature of it's CV3.0-content on this work.

Additionally, a small client for the PoC will demonstrate the validation of the digital signature on the side of the inquiring party. Implementation will be provided as standalone application or integrated as web-application in the content delivered by the `cv3cas`.

The keys used in signing are delivered as part of the CV3.0-content by making use of the cert-ontology [35].

10.5 CV Instantiation

At a given point in time, a new CV has to be instantiated in the `cv3cas`, two different approaches can be taken:

- Create a new CV3.0 as soon as the owner of the CV3.0 logs in for the first time

- Periodically create CV3.0s as a batch process

In the first case, upon initial login of the user, documents and attributes from all organisational facilities are retrieved and the CV3.0 is created. In the second case, a periodical batch job (for instance running at the end of each semester or year) gathers all data from the organisational facilities for a specified subset of CV3.0-owners.

It currently has to be determined, which possibilities exist in accessing the organisational facilities at BUAS for the PoC. The possibilities given determine the implementation of the CV3.0-instantiation.

References

1. BFH Homepage, http://www.bfh.ch

2. W3C - RDF Primer, http://www.w3.org/TR/rdf-primer

3. Wikipedia - Transport Layer Security, https://en.wikipedia.org/wiki/Transport_Layer_Security"

4. Wikipedia - Document Management System, https://en.wikipedia.org/wiki/Document_management_system

5. Wikipedia - Identity Management, https://en.wikipedia.org/wiki/Identity_management

6. Wikipedia - Federated Identity Management, https://en.wikipedia.org/wiki/Federated_identity_management

7. Wikipedia - Directory Service, https://en.wikipedia.org/wiki/Directory_service

8. SWITCH Authentication, https://www.switch.ch/aai

9. SuisseID Homepage, http://www.suisseid.ch

10. git Homepage, http://www.git-scm.com

11. git - Git Objects, https://www.git-scm.com/book/en/Git-Internals-Git-Objects

12. pm - CV3.0, https://denethor.bfh.ch/projects/iam-group-cv3-0

13. Stackoverflow - RFC3161 Timestamps, http://stackoverflow.com/questions/11913228/how-can-i-use-rfc3161-trusted-timestamps-to-prove-the-age-of-commits-in-my-git

14. Node.js Homepage, http://nodejs.org

15. Apache Jena - Fuseki, https://jena.apache.org/documentation/serving_data/

16. W3C - Triple Store, http://www.w3.org/2001/sw/Europe/events/20031113-storage/positions/rusher.html

17. University of California - REST, https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

18. W3C - HTTP, http://www.w3.org/Protocols/rfc2616/rfc2616.html

19. W3C - Content Negotiation, http://www.w3.org/Protocols/rfc2616/rfc2616-sec12.html

20. W3C - Turtle, http://www.w3.org/TR/turtle/

21. RDF+XML, https://tools.ietf.org/html/rfc3870

22. JSON Homepage, http://json.org

23. JSON-LD, http://json-ld.org/spec/ED/json-ld-syntax/20120522

24. Uduvudu, http://uduvudu.me

25. SWITCH - swissEduPersonUniqueID, https://www.switch.ch/aai/support/documents/attributes

26. W3C WebID Community Group, http://www.w3.org/2005/Incubator/webid/wiki/Main_Page

27. BFH WebID Identity Provider, http://webidp.bfh.ch

28. See section 8.3.2, Unique Identifier

29. See section 8.3.5, User Consent

30. W3C - Web Access Control, http://www.w3.org/wiki/WebAccessControl

31. Wimmics - Shi3ld, http://wimmics.inria.fr/projects/shi3ld/

32. Signing RDF Graphs, http://www.hpl.hp.com/techreports/2003/HPL-2003-142.pdf

33. Web Payments, http://payswarm.com

34. GitHub - Payswarm Client, https://github.com/digitalbazaar/payswarm.js/blob/master/lib/payswarm-client.js

35. W3C - Cert Ontology, http://www.w3.org/ns/auth/cert