

Data Exchange in Practice: Towards a Prosopographical API

Georg Vogeler, Gunter Vasold, Matthias Schlögl

Universität Graz / Österreichische Akademie der Wissenschaften, Universität Graz, Österreichische Akademie der Wissenschaften
georg.vogeler@uni-graz.at, gunter.vasold@uni-graz.at, matthias.schloegl@oeaw.ac.at

Abstract

The paper discusses the question whether methods applied in work with prosopographical data can be integrated into an "International Proposography Interoperability Framework" (IPIF) comparable to the IIIF (International Image Interoperability Framework). It suggests basing this upon a RESTful interface defined in a publicly available definition of an API. The API is based on the three-partite factoid model, thus keeping the identification of the person, information on the person, and the source documenting this information separated, aggregating it in a "factoid" with the metadata of its creation. The API definition follows the rules of OpenAPI which allows automatic code generation. As a proof of concept, the API has been implemented in the context of the APIS project, containing the information of the Austrian biographical lexicon, and with data from monasterium.net, the world wide largest database of medieval charters. It introduces a wrapper for LOBID and wikidata and a standalone server created with code generation from the OpenAPI specification as a second proof of concept implementation. In a third proof of concept it showcases the reuse of tools originally developed for APIS in other services.

Keywords: RESTful API, OpenAPI, factoid model, APIS, monasterium.net, IPIF, Data aggregation, prosopographical resources.

1 Introduction

This paper discusses whether methods applied in work with prosopographical data can be integrated into an "International Proposography Interoperability Framework" (IPIF) comparable to the IIIF (International Image Interoperability Framework). Prosopography is a field of research in which biographical information on a selected population of individuals are aggregated to detect patterns in relationships, careers, and other attributes relevant for social, cultural, or political observations (Charle, 2015). This field of study can draw on a wide range of resources, ranging from scholarly publications of historical sources to dedicated prosopographical databases, processed with a wide range of methods - from natural language processing to network analysis - and it has a rich history of research, in which the digital transformation took place early (Althoff, 1976; Imhof, 1978; Bulst, 1989; Goudriaan, 1995; Keats-Rohan, 2007). To bring this "digital prosopography" forward, we take up on earlier proposals to create APIs for prosopographical data (Ebneith and Reinert, 2017) and suggest a shared RESTful API definition to facilitate communication between prosopographical data resources, web front ends, and analytical tools. This is motivated by the identification of common use cases and the need for an efficient technical infrastructure. This paper will start with a description of the motivation (section 2), describe the current proposal in detail (section 3), and demonstrate its feasibility in three prototypes (section 4).

2 Motivation

2.1 Common Use Cases

An IPIF must be based on concrete application scenarios. A major use case of the IPIF is an aggregated query on several services. Whenever a humanities scholar mentions a person, the easy look-up of existing information on the person is a standard requirement. The users want to look up and

reuse information on a person, of which they have some basic information (usually a name) available. Information on this person can be provided by a wide range of resources: authority files (usually aggregated into VIAF); general-purpose databases like Wikidata; general prosopographical databases like those created from ancient and medieval sources (e.g. DPRR, PASE); or specific prosopographical databases documenting a group like clergy (e.g. CCed); parliamentarians (e.g. BIOPARL) or artists (e.g. Bayerisches Musiker-Lexikon Online. <http://bml.o.de/>); biographical dictionaries in prose like the wide range of national biographical dictionaries of which (Fox, 2019) gives the most recent overview, or community efforts by family historians. Rich information is available in resources not even dedicated to prosopographical information like scholarly editions, as in tax registers (for example the Hearth Tax series by the British Academy¹) or letters, as for instance aggregated in the [correspSearch](http://correspsearch.net) service², or results of automatic text annotation, for instance in OCRed newspapers from projects like NewsEye³. In the scenario the users would use an application with a search parameter and list of URLs of arbitrary IPIF endpoints. This application can be a GUI with a search slot or part of an automatic information linking architecture. Typically, the user would use this application to search for the name at question and resolve the name to matching authority IDs. This functionality could be included in any kind of editing and annotation process, e.g. tools like Pelagios/Recogito⁴ could be extended beyond geographical gazetteers, or tools for scholarly editing like [ediarum](http://www.bbaw.de/telota/software/ediarum)⁵ could extend TEI-P5 person markup beyond the link to a local databases.

¹<http://gams.uni-graz.at/context:htx>

²<https://correspsearch.net/>

³<https://www.newseye.eu/>

⁴<https://recogito.pelagios.org/>

⁵<http://www.bbaw.de/telota/software/ediarum>

We consider the reuse of basic analytical tools as the second major use case. Networks of connections between persons or lists of events in the life of a single person are analytical tools applied in many projects. Most prosopographical resources offer network visualisations, e.g. ego-networks on a single entry or networks of a selected population⁶. The second analytical use case is the display of chronologically sorted structured information on a single person. IPIF will facilitate interfaces creating these common visualizations from a single resource or by aggregating information across distributed resources. Another common usage scenario is the enrichment of prose. Several resources add structured information to biographical prose: examples include Wikipedia infoboxes⁷, introductory notes in biographical databases, and the standard display of cosmotool⁸. Again, IPIF will facilitate the creation of these enrichments from a single resource as well as aggregating it from distributed resources. These functionalities do not rely on modelling decisions affected by specific research questions or data sources but reflect common prosopographical use cases.

There are other use cases which could be addressed by IPIF, although they might be less obvious and much more dependent on individual modelling decisions. We would like to summarize them under the headings "Biographical Dictionary," "Fact Checking," "New Interpretation," "Database Publication," but assume that this list can be easily extended. Let us describe some of them briefly: The author of any kind of biogram ("Biography Dictionary"), e.g. in a biographical dictionary, would profit from a search interface aggregating textual fragments from primary (and secondary) sources including the reference. The "New Interpretation" scenario adds new interpretations of existing sources on a person, while "Fact Checking" looks for justifications of statements about a person in a set of digital resources. These scenarios depict both research with prosopographical data as well as the generation of new data. The definition of the API has been influenced by a group of scholars working on religious orders⁹. During a workshop held in 2017 in Vienna the group discussed how to use/improve the API for the edition of primary resources.

⁶E.g. the online portal of "Deutsche Biographie": <https://www.deutsche-biographie.de/graph?id=sfz53095>

⁷<https://en.wikipedia.org/wiki/Help:Infobox>

⁸<https://cosmotool.de.dariah.eu/>, an example for the gives https://cosmotool.de.dariah.eu/cosmotool/person/Wikidata_Q5879

⁹Participants were Thomas Wallnig (Vienna), P. Alkuin Schachenmeyer OCist (Klosterneuburg), Irene Rabl (Vienna), Hedvika Kuchařová (Prague), Jana Borovičková (Prague), Miguel Vieira (London), James Kelly (Durham), Nada Zečević (London), Daniel Jeller (Vienna), Stephan Makowski (Cologne), Ekaterini Mitsiou (Vienna), Christian Popp (Göttingen), Stefan Eichert (Vienna), Bärbel Kröger (Göttingen), Gunter Vasold (Graz), Georg Vogeler (Graz) representing projects like CCEd, Germania Sacra, Pez-Correspondence, Jesuit Networks, "Who were the nuns?", PRODOMO (<http://prodomo.icar-us.eu/>)

2.2 Efficient technical infrastructure

The second major motivation for the proposal of IPIF is the lack of an efficient technical infrastructure to serve these use cases. We consider the Web of Data activities of the W3C to be the best conceptual approach to publishing databases and for making individual small amounts of data queryable on the web. Data publication in RDF is widely used and can be considered a well-supported technology. For the programmer's access to these data publications, the W3C proposed to make the RDF data not only available as data dumps but also via a query API, the SPARQL protocol¹⁰. The SPARQL protocol defines a RESTful API which allows submitting queries in a common query language (SPARQL) and receiving structured data in return. SPARQL is designed to cover arbitrary levels of complexity. The API is therefore completely agnostic to the data queried.

This creates problems for the envisaged use cases: First, the use cases and the power of a full SPARQL endpoint are not balanced. While the use cases can be served by simple and efficient queries, the technical maintenance of a SPARQL endpoint is very demanding. As a SPARQL endpoint is designed to process any kind of query possible, (computationally-)expensive or even harmful queries are possible, e.g. requesting huge amounts of data or filtering on a large dataset without making use of indices.¹¹ This and the lack of native permission/user management specifications in SPARQL are arguments for system administrators to prefer not to offer a SPARQL-endpoint. Even big institutions providing central data resources to the public such as the German National Library (GND) decided not to maintain a SPARQL endpoint for their data.

Second, querying an RDF resource does require fundamental knowledge of SPARQL, and a deep understanding of the ontology used. Wikidata, for instance, uses a set of more than 6,000 RDF properties.¹²

While SPARQL offers a standardized interface to any triple store, it does not solve the problem of ontology alignment. That means that a useful federated query is only possible across triple stores that use known ontologies or map to a common ontology. In prosopography, the attempts to map data to CIDOC CRM in the data4history consortium seem the most promising, but have not yet found sufficient implementations.

Several approaches exist to solve the technical problems created by SPARQL endpoints, like moving parts of the complexity of the SPARQL request into the client via dumps; smart brokers between requests and SPARQL server like Linked Data Fragments (Verborgh et al 2016)¹³; or data virtualization services, like OpenLink Virtuoso uri-burner.¹⁴ Other approaches hide the SPARQL endpoint behind a RESTful API that transforms API requests into pre-

¹⁰<https://www.w3.org/TR/sparql11-protocol/>

¹¹The SPARQL FILTER() procedure is applied only after materialisation of the graph selected by the graph pattern in the query

¹²https://www.wikidata.org/wiki/Wikidata:Database_reports/List_of_properties/all, 2019-08-29.

¹³<https://linkeddatafragments.org/>

¹⁴<http://uriburner.com/>

defined SPARQL queries like grlc (Meroño-Peñuela and Hoekstra, 2016) or the recently published LOBID service to GND data provided by the “Hochschulbibliothekszen- trum des Landes NRW”.¹⁵

Digital humanities practice seems to prefer these interme- diate solutions. Tools like openRefine Reconciliation use Wikidata for our first use case efficiently. OpenRefine fol- lows the path of hiding the SPARQL-Endpoint behind a restricted API focussed on the necessary information for reconciliation.¹⁶ In particular the publication of resource- specific RESTful APIs is establishing itself as alternative technology in the digital humanities. They allow far-less- flexible queries compared to SPARQL endpoints, but are much more stable and can be implemented more easily. RESTful APIs are a quasi-industry standard and are sup- ported by all web development frameworks and program- ming languages. Several prosopographical databases offer their data not via SPARQL endpoint but via their own API: the Deutsche Biographie, for instance, reuses the SOLR- API;¹⁷ the database of persons provided by the Germania Sacra project has defined its own API;¹⁸ and the above- cited correspSearch service¹⁹ offers an API based on the TEI models used inside the service which is a similar ap- proach to the one taken by the large digital scholarly edition of all documents related to the Composer Carl Maria von Weber.²⁰

All these solutions solve the technical problems of the SPARQL endpoints, but cannot solve the problem of har- monising the query methods, results or data models. They usually only provide verbal descriptions of the service. With OpenAPI (formerly swagger) and Core API, there are proposals to describe such API definitions in a standard- ized way. These technologies allow for an easy implemen- tation of APIs and—maybe perhaps more importantly—for a (semi-)automated creation of clients.

This approach extends established practical solutions like the BEACON format for identifier alignment²¹, which has been promoted by the Wikipedia and is adopted by many prosopographical data resources. While BEACON defines just a data format, we propose a dynamic solution allowing interactions with the data source.

We conclude from the existence of common use cases and advantages of lightweight RESTful APIs compared to SPARQL endpoints, that creating a public definition of a shared RESTful API is a valid path for prosopographical

data aggregation and processing. A shared API dedicated to prosopographical data should facilitate the implemen- tation of applications for the above-mentioned use cases and make the data practically interoperable. To achieve this, the API definition has to propose a flexible but efficient data model and methods to cover the core use cases described above.

3 The API

In the following section we will give a descriptive introduc- tion into the API. A formal description according to stan- dards OpenAPI is available on github.²²

3.1 The data model

The list of possible sources given in section 2.1 might al- ready demonstrate that there is information on different lev- els of description available which leads to different data models. This model is the result of workshops with proso- pography experts and developers held in Vienna (2017, 2019), and in the data for history-group in Leipzig 2019,²³ which identified the diverging data models for the individ- ual statements about individuals as the core problem of a real data exchange. (Fokkens and ter Braake, 2018) have already discussed the variety of data models and suggest a repository with data models and examples to help users to understand and reuse existing models. They identify the BIO-CRM proposed by (Tuominen et al., 2017) as a possi- ble overarching model. The data model of IPIF therefore tries to be aligned with this model.

Nevertheless, IPIF does not attempt to describe a concep- tual model covering the full range of prosopographical use cases and data sets. In fact, the scenarios described above do not require full harmonisation. The analytical tools dis- cussed in section 2.2 need a simple common data model. It has to cover relationships between people (“Social Net- work”) and dateable information about a person (“Career”). Therefore, IPIF is not meant to deliver the whole richness of data that might be available in projects. For interoper- ability on the ontology level there exist several approaches such as schema.org,²⁴ CIDOC CRM,²⁵ the Europeana Data Model,²⁶ etc. The model presented below has to be distin- guished from these efforts to tackle interoperability issues in the digital humanities. IPIF is not meant to replace ex- isting data models, but rather to be an easy path to access existing data for common use cases that simply require a downsized version of the data.

Still, the development of data models in prosopography has led to one common concept beyond the use cases de- scribed above. IPIF realizes a conceptual model for which

¹⁵<https://www.hbz-nrw.de/produkte/linked-open-data>

¹⁶<https://github.com/OpenRefine/OpenRefine/wiki/Reconciliation-Service-API>

¹⁷<http://data.deutsche-biographie.de/about/#solropen>

¹⁸<https://adw-goe.de/forschung/forschungsprojekte-akademienprogramm/germania-sacra/schnittstellen-und-linked-data/>

¹⁹<https://correspsearch.net/index.xql?id=api&l=en>

²⁰https://weber-gesamtausgabe.de/en/Help/API_Documentation.html

²¹<https://de.wikipedia.org/wiki/Wikipedia:BEACON>

²²<https://github.com/GVogeler/prosopogrAPI>, This paper describes the sta- tus of the proposal in commit <https://github.com/GVogeler/prosopogrAPI/commit/7924bd513980bd776a34761806b3d6e63d99e2f5>

²³<https://github.com/GVogeler/prosopogrAPI/commit/7924bd513980bd776a34761806b3d6e63d99e2f5>

²⁴<http://schema.org>

²⁵<http://www.cidoc-crm.org/>

²⁶<https://pro.europeana.eu/resources/standardization-tools/edm-documentation>

(Bradley and Short, 2005) introduced the term factoid. A factoid is formed by three main information units: statements about an individual person justified by the source. This factoid is the result of an interpretation of the source by a researcher at a specific time, and can be considered as one instantiation of generic models to represent perspectives on facts (Fokkens and ter Braake, 2018). John Bradley used the conceptual model in several projects at King's College London (PASE, DPRR, CCEd). The generality of the model is proved by independent realisations of the model in other prosopographical resources: The Repertorium Academicum Germanicum uses a similar three-part model (Baeriswyl-Andresen, 2008). The Personen-datenrepositorium (PDR) of the Berlin-Brandenburgische Akademie der Wissenschaften (Neumann et al., 2010) uses the same conceptual model and by this the model spread into projects based on the PDR, e.g. MusMig²⁷, or the Jesuit Science Network²⁸.

This model deviates from the personal databases that have established themselves as controlled vocabularies and references for Linked Data in the Digital Humanities. These usually do not take into account the process of aggregating information about a person from historical sources. Also, the approach thus deviates from the TEI's "personography",²⁹ which, like the linked data resources, describes a person with a list of characteristics, although TEI could be used to express the model, as the syriaca.org project has recently demonstrated (Schwartz, 2019).

There are several attempts to transfer the factoid conceptual model into RDF. (Pasin and Bradley, 2015) proposed a CIDOC CRM based version of the Factoid model and published corresponding ontologies.³⁰ The conceptual model has also been realized with other vocabularies in SNAP, which re-uses vocabulary from the Linking Ancient-Wisdom project (Lawrence and Bodard, 2015). The King's Digital Lab has recently published the prosopographic database on the Roman Republic as a LOD resource including a SPARQL endpoint using its own ontology to describe the Factoids.³¹ Wikidata offers a data model to identify contributions as claims made by a contributor (Erxleben et al., 2014) and therefore can be considered as another proposal to implement the factoid conceptual model in RDF. Finally, the W3C has proposed a vocabulary similar to the needs of the factoid model: The provenance data model (PROV, (Missier et al., 2013)) covers relationships between data and sources. In the PROV model the creation of a factoid is the activity (`prov:activity`) connecting the source (`prov:used`) and the resulting statement (`prov:entity`). (Ockeloen et al., 2013) have applied this to biographical data. You can interpret the factoid also as an annotation on the source and express it with

W3C Web Annotation.³² The IPIF statement maps to the annotation body, the source to the target in the Web annotation data model. Both cover mainly the relationship between source and information. In fact, even RDF reification and named graph specifications could be used to implement factoid structures, mapping the IPIF statements directly to RDF statements.

Therefore, the model proposed by IPIF is a very simple one: IPIF suggests a resource called *factoid*, which is the aggregation of the three objects: the combination of *statements* on a *person* justified by a *source*. As the factoid is created by a scholar or an algorithm, it has to contain metadata on the responsible person and date of creation (`createdBy`, `createdWhen`) and later updates (`modifiedBy`, `modifiedWhen`). The person resource needs no more properties than a local id (`@id`) and alternative URIs (`uris`). The source-object carries a descriptive string (`label`), a local id (`@id`) and alternative URIs (`uris`) as attributes. The statements (`statement`) are the most complex entity in the model as they should cover main information structures in prosopographical descriptions.

The data model of the statement is loosely based on an event model as every statement can include a temporal information. However, following the proposal by (Tuominen et al., 2017) for a Bio-CRM, it allows for unary and binary properties as well. IPIF suggests using a `statementContent` property to deliver a verbal description, e.g. a phrase extracted from a source or a short biogram. This is the most generic property of a statement and serves as a fallback for all information in an existing database which cannot be mapped to other IPIF properties. Usually, the content can be described in a more structured way. The main scenarios given in section 2.1 suggest two types of information: the relationship to other persons and events. The first is realised in the `relatesToPerson` property. Additionally a social relationship can be created by the membership in a group or institution, for which IPIF uses the property `memberOf`. The second scenario is realised in a set of properties of an event in the biography of the person: Any event related to the person can be described by the role (`role`) of the person in the event, and its temporal (`date`) and geographical (`place`) allocation. The statement can describe the type of the event (`statementType`). The samples collected by (Fokkens and ter Braake, 2018) suggest considering additional properties to model lifespan, gender, and "occupation/claim of fame/person-type." From the experience in APIS it has been suggested to include creative and productive activities with a dedicated property `contributesTo`.

It should be noted that there is no explicit event object. This is for two reasons, the first pragmatic: many existing data structures (e.g. XML/TEI-P5 or simple csv tables) share this flat structure of the descriptive properties of a person. The model allows for easy mapping of these resources. The second is that the event model is implicit in most of the information on a person, so it can be inferred from the current data model. Every statement can be converted into the description of an event by simply adding a date. Conceptu-

²⁷<http://www.musmig.eu/home/>

²⁸<http://jesuitscience.net/>

²⁹<https://www.tei-c.org/release/doc/tei-p5-doc/en/html/ND.html#NDPERS>

³⁰<http://factoid-dighum.kcl.ac.uk/> and <https://github.com/johnBradley501/FPO>

³¹<http://romanrepublic.ac.uk/rdf/doc/ontology.html>

³²<https://www.w3.org/TR/annotation-model/>

ally one could even argue that every statement is implicitly rooted in time, although sometimes with unknown dates. In practice, many users do not care for this unknown information. The practical advantages of this decision become clear when we consider the “Social Network” analysis scenario. Any kind of social network needs two persons as nodes connected via an edge. The data model covers this scenario efficiently by filtering all statements on a person with the property `relatesToPerson`. Still, event-based modelling suggests that most of these relationships are not stable, but could change in time. Even the biological relationship to parents is established by the birth. The model represents this by making a statement possible which includes relationship information and a date.

Practical considerations lead to an additional explicit property, the name as the main identifier of a person in spoken language. Again, the advantage of the implicit event-based model becomes clear considering the double use of names as general identifiers and as appellations which can change in time. Binary statements relating the person to others, like family relationships, are covered by the `relatesToPerson` property. The relationship can be specified using the `role` property. Unary statements like “main occupation” or “gender” are covered by a combination of the `role` property and `statementType`. The `role` property contains the predicate of this unary triple like statements, and `statementType` the object. Thus, they can be considered to be a first fallback for parts of the source model which do not map directly to the IPIF data model.

Most of the properties named above are defined as objects which can be described by an URI (`uri`) as well as by a human readable label (`label`). This model applies to roles (`role`), places (`place`), relationships to persons (`relatesToPerson`), and group memberships (`memberOf`), and categories (`statementType`).

We expect that API providers use controlled vocabularies for the URIs, e.g. the TEI guidelines on personography,³³ Bio-vocab³⁴, FOAF³⁵ (Brickley / Miller 2014), schema.org,³⁶ or Swiss Art Research Project (SARI).³⁷ IPIF suggests referencing these vocabularies in the describe-Resource (`/describe/vocabs`). Only the name and the date are properties of the statement-object with specialised models: the date can be described by a verbal label and a formalized `sortDate` in W3C schema format (`xs:date`), while name is just a string.

The factoid is the aggregation of the three objects, i.e. the combination of statements on a person justified by a source created by person. Therefore, it adds metadata on the responsible person and date of creation (`createdBy`, `createdWhen`) and later updates (`modifiedBy`, `modifiedWhen`).

³³<https://www.tei-c.org/release/doc/tei-p5-doc/en/html/ND.html#NDPER>

³⁴<http://vocab.org/bio/>

³⁵<http://xmlns.com/foaf/spec/>

³⁶<https://schema.org/Person>

³⁷<https://docs.swissartresearch.net/et/person/>

3.2 Endpoints and parameters

The data model is reflected in the main endpoints of the API: `factoids`, `statements`, `persons`, `sources`. The `sources` endpoint returns bibliographic or archival identification of the historical sources in which the information on a person is based. This can be a verbal description (`label`) and list of URIs (`uris`) pointing to further information. The resource returns, as all the resources of the main endpoints, the local ID (`@id`) and metadata on the creation and modification of the entry. IPIF suggests using pointers to machine-readable resources, e.g. other RESTful APIs. Via the `persons` endpoint the API consumer can retrieve identification information on the individuals. This endpoint can be addressed by the local id or any other id stored for the same person, in particular URIs from authority files. It returns nothing else but the local id (`@id`) and a list of alternative URIs (`uris`). The `factoids` endpoint aggregates the information on source, person, statements made, and the creation of the statements by a researcher as metadata to the factoid.

All endpoints allow querying by parameters stored in resources linked to the current resource. The API defines for this parameters like `statementId` (return all resources which are related to an identified statement), `sourceId` (return all resources related to source with the given id), `factoidId` (return all resources related to a factoid with the given id), and `personId` (return all resources related to a person with the given id). This makes quick traversals through the prosopographical database possible. With an API call like `/statements?personId=ia14328` you can receive all statements made upon an individual, e.g. as to prepare a timeline by sorting the results with a date property.

In addition to the ids of resources, this generic search pattern includes full-text search in all values stored as property of the resource. `?st` filters by a full text search in all properties of a statement related to the current resource, `?f` filters by full text search in the all properties resources connected to the current resource via a factoid, and `?p` filters the current resource by a search in the properties of the person object. `/person?st=Bishop` returns all identifiers of persons, on which statements exist containing the word “Bishop.” The precise functionalities of the full text search are a decision of the API provider. IPIF expects to handle URIs as single words, that a search `/factoids?p=http%3A%2F%2Fviaf.org%2Fviaf%2F289779` returns all factoids on Engelbert I, archbishop of Cologne, as identified by his VIAF URI as stored in the person object’s `uris` property.

As described above, the `statements` have the richest data model. Therefore, statements can be queried not only by id and a full-text query over all properties, but by single property values as well, i.e. `relatesToPerson`, `place`, `memberOf`, `name`, and `role`. Those properties carrying a URI as descriptor can therefore be used for linked data queries as the URIs stored for related person, place and even statement contents from a controlled vocabulary.

Temporal searches are facilitated by `from` and `to` query parameters. A special interpretation mode is de-

ned to handle fragmentary data: Fragments (yyyy, yyyy-mm) will be interpreted as exact time ranges if the second parameter is missing (from=yyyy-mm is interpreted as from=start of month, to=end of month). If conflicting data is present, for example from=yyyy&to=yyyy-mm-dd, the most correct interpretation will be decided on by the backend. For instance, the example above will be interpreted as from=yyyy-01-01&to=yyyy-mm-dd.

Paging of filter results is supported by parameters for page size (size), page number to be returned (page) and sorting (sortBy).

As applications might only want to traverse via a list of results, the API provider is asked to return only the id of the resource matching to search parameters and the property searched. This will in particular facilitate autocomplete services like “type in a part of a name and the application will list all the persons to which a factoid with the name-statement containing this part exists,” where selecting an entry can trigger other requests based on the id.

In the current definition of the API, the resources are returned as JSON objects, in which lists are represented as arrays and properties allowing a verbal and a formal representation as objects.

It has to be noted, that in combination with the REST endpoint to the respective resource (e.g. {server}/{api}/person/{id}), the local ids constitute globally unique URIs reusable as RDF resource identifier. In fact, the responses of the API can be interpreted as RDF, e.g. by adding a context description to the JSON response, for which a first draft can be found on github.³⁸

A fifth resource returns metadata on the data provider itself: /describe gives basic information about the service and the implementation of the API. This includes information on the service provider (provider, contact), a verbal description of the service (description), and in particular a statement on the level of compliance to the API definition (complianceLevel).

To facilitate the creation of REST interfaces implementing the suggested API as front ends to existing databases, each end point has to provide information about the supported API version(s) and compliance levels. The API is organized in 3 levels of compliance. The supported compliance level of a specific server can be queried via an API call to server/api/describe. Based on this information a consuming program can find out which requests are supported by the server. On the server side, compliance levels allow decisions about how much effort is to be invested in the development of an API-compliant interface. Implementing level 0 will be easy and can be done without much effort; level 1 is a bit more complicated; level 2 will require more development work or the usage and/or adaption a third party software like the one which will become available as reference implementation.³⁹

Compliance level 0 is a minimalistic implementation which only supports HEAD and GET requests. It can be used with

³⁸<https://github.com/GVogeler/prosopogrAPI/blob/master/context.json>

³⁹<https://github.com/GVasold/papilotte>

all parameters supported by these two methods with the exception of the fulltext searches p=, st=, s= and f=. Implementing this level should be easy, as it only maps API calls to rather simple database queries. Data providers can extend existing services by this functionality or integrate a dedicated server software. We are working on a reference implementation for the API which can act as a proxy to an existing database. This server will support pluggable adapters in form of custom simple functions which implemented these mappings to the database, while the rest of the software can be used unchanged.

Compliance level 1 also only supports read-only access (HEAD and GET), but additionally allows filtering by fulltext searches over multiple fields, so implements all parameters for these two methods as documented in the API specification. Again this level can be added to existing services or can be provided via a proxy like the reference implementation as described above.

Compliance level 2 fully implements the API which extends level 1 with the possibility to add and modify data. Accordingly, level 2 is only useful for clients which add or manage data on a server. As writing data to existing databases heavily depends on data models which will normally not fit the IPIF data model, this will mostly be useful for new projects settled on the IPIF model, for projects which want to (cyclically) export data from their main database to provide it via a dedicated IPIF server, or for collecting and integrating data from different sources into a single database. The reference implementation will support level 2 and therefore the described use cases.

4 Prototypes

The API is currently deployed in three prototypes as proof of concept. The first is built upon an existing prosopographical database: APIS - Austrian Prosopographical Information System.⁴⁰ APIS is a research project financed by the Austrian “Nationalstiftung.” Its ultimate goal is the semantic enrichment of the Austrian Biographic Dictionary (ÖBL).⁴¹ During the project, a system was developed at ACDH that turned into a generic tool for storing and curating prosopographical data, called the A(ustrian) Prosopographical Information System (APIS).⁴² A central idea of APIS is allowing automatic tools and human researchers to work on the same data. Therefore, it was and still is essential to provide access to the data via APIs. APIS currently features two different APIs: a hyperlinked API that delivers atomic data and allows the retrieval of fine-grained information, and a second API that delivers everything that APIS holds on a specific entity. For the latter, APIS uses so-called “renderers” to convert the original Python objects coming from the serializer into various formats. Currently, APIS supports a full-featured json serialization using internal attributes, a very basic XML/TEI serialization, a basic

⁴⁰<http://apis.acdh.oeaw.ac.at/>

⁴¹The ÖBL is a national biography. It is written at the Austrian Academy of Sciences since the 50s and contains roughly 20.000 biographies of people who had an impact on what was then Austrian soil.

⁴²<https://github.com/acdh-oeaw/apis-core>

CIDOC CRM serialization (in various formats), and the serialization to the prosopography API format.⁴³ It has to be noted that the APIS reference implementation at the current stage only offers compliance level 1 of the IPIF definition (GET requests via identifier).⁴⁴

A second prototype has been implemented with data from the *monasterium.net* charters database transferred into a stand-alone IPIF server. It uses a data set collected in the context of the FWF project “Illuminated Charters”⁴⁵ which contains a rich set of names on cardinals issuing collective indulgences.⁴⁶ *Monasterium.net* is an eXist-db instance using TEI und CEI schemata⁴⁷ for documents and prosopographical data. The prosopographical data follows the proposal of the TEI “personography”⁴⁸ with simple properties (<name>, <occupation> with <date> for the period of activity) of a person referenced in the charter description as a <persName >. The data was exported from the *monasterium* XML database to a JSON representation following the proposed IPIF model using the XQuery language. The data has been used as a first test for *Papilotte*, a flexible and extensible IPIF server currently being developed at the Centre for Information Modelling of the University of Graz.⁴⁹

Papilotte uses the *Connexion* library,⁵⁰ a framework which handles HTTP requests based on a given OpenAPI specification. This means that *Connexion* reads-in the IPIF OpenAPI specification and routes paths defined in the specification to Python functions. It also relieves the programmer from annoying tasks such as validation of incoming and outgoing data against the schemata defined in the specification, validation and casting of request parameters, exception handling, authorization and the like. *Connexion* is a good example of how an OpenAPI specification can speed up and facilitate the development of software based on this API specification. Similar tools are available for many programming languages and can also be used for data consuming client applications.

The core functionality of *Papilotte* is to provide an IPIF conform REST interface, implementing the full functionality of IPIF on compliance level 2. *Papilotte* is very flexible in terms of data sources because data is provided via configurable connectors based on a simple abstract connector

⁴³Please see <https://gist.github.com/sennierer/d0dc2311b36a00d0a88541e27988ae43> and <https://apis.acdh-dev.oeaw.ac.at/apis/api2/entity/43670/?format=json%2Bprosop> for an example biographies serialized in the prosopography API format.

⁴⁴The APIS internal APIs can of course be used to query for the entities.

⁴⁵<https://illuminierte-urkunden.uni-graz.at/>

⁴⁶<https://www.monasterium.net/mom/index/BischoefeAblaesse>

⁴⁷<http://www.cei.lmu.de>

⁴⁸<https://www.tei-c.org/release/doc/tei-p5-doc/en/html/ND.html#NDPERSE>

⁴⁹<https://github.com/gvasold/papilotte>, the implementation is currently available at <https://ginko.uni-graz.at/illurk/api/ui/>

⁵⁰<https://github.com/zalando/connexion>

class. Connectors for JSON-files and relational databases including a built in SQLite database are part of the software or will be added as soon as the API has settled. Adding custom connectors to other data sources is as simple as implementing a single class for each type (factoid, person, source and statement) implementing three methods: `get()`, `search()` and `count()`. Each method queries the data source and returns the result in a format defined by the IPIF specification. *Papilotte* takes care of the rest.

Custom connectors can be written to access any form of data source, for example existing relational databases, document stores like MongoDB, XML databases, graph databases or triple stores. Connectors can also act as proxies to web services like SPARQL endpoints or custom APIs to make their data easily accessible via the IPIF API.

The third proof-of-concept implementation demonstrates how IPIF could be used to deliver potential entity candidates. The application acts as a middle layer between SPARQL endpoints and Rest APIs on the one side and the IPIF API on the other. Requests posted to the IPIF API are transformed to SPARQL or Rest API GET queries and forwarded to the respective APIs.

The application includes an admin backend to configure the layer. Two steps need to be configured: the request and the return. The configuration of the request uses a Python format string and ingests IPIF query attributes in the string. The formatting of the return json utilises the power of the Django html templating engine.⁵¹ Whatever is returned from the queried APIs is ingested in these templates. The templating engine allows for simple processing such as for loops, if statements, concatenating, slicing, and others directly from within the template. The proof of concept application was configured for querying LOBID⁵² and wikidata.org.⁵³ Results from both APIs are provided as-is without any disambiguation. Figure 1 shows the date of birth statements of two returns for the query `/person?f=Kreisky`. The left snippet was returned from LOBID, the right from Wikidata. Both entries can be matched via the list of uris also returned from the APIs and seamlessly merged in a richer entry.

Finally, the proof of concept of the IPIF can be demonstrated by a prototypical reuse of a simple network visualisation application. The application was developed for the APIS project.⁵⁴ It allows the iterative creation of network visualizations between entities (e.g. person-place, person-person) and export the final network to graphml or a json format for further analysis in tools such as Gephi. The application was refactored to read the data from the IPIF API. The application was able to read the data from the MOM-

⁵¹<https://docs.djangoproject.com/en/2.2/topics/templates/>

⁵²LOBID is a service provided by “Hochschulbibliothekszentrum des Landes NRW” serving the GND via a Rest API: <https://lobid.org/gnd>

⁵³Via the SPARQL endpoint

⁵⁴See <https://bit.ly/2kNfSiC> for the source code and <https://pmb.acdh.oeaw.ac.at/apis/entities/networks/generic/> for a live version of the visualization tool

```

"person": {
  "@id": "170686299",
  "uris": [
    "http://d-nb.info/gnd/170686299/about",
    "http://viaf.org/viaf/203136754",
    "http://www.wikidata.org/entity/Q1322819",
    "https://de.wikipedia.org/wiki/Peter_Kreisky"
  ]
},
"source": {
  "@id": "GND",
  "metadata": "GND via Lobid API"
},
"statements": [
  {
    "@id": "Stmt170686299_preferredName",
    "name": "Kreisky, Peter"
  },
  {
    "@id": "Stmt170686299_dateOfBirth",
    "date": {
      "label": "1944",
      "sortdate": "1944"
    },
    "role": {
      "label": "dateOfBirth"
    }
  },
  {
    "@id": "Stmt170686299_dateOfDeath",
    "date": {
      "label": "2010",

```

```

"person": {
  "@id": "Q1322819",
  "uris": [
    "http://d-nb.info/gnd/170686299",
    "http://viaf.org/viaf/03136754"
  ]
},
"source": {
  "@id": "Wikidata",
  "label": "Wikidata via Sparql"
},
"statements": [
  {
    "@id": "StmtQ1322819_preferredName",
    "name": "Peter Kreisky"
  },
  {
    "@id": "StmtQ1322819_dateOfBirth",
    "date": {
      "label": "1944-05-08T00:00:00Z",
      "sortdate": "1944-05-08T00:00:00Z"
    },
    "role": {
      "label": "dateOfBirth"
    }
  },
  {
    "@id": "StmtQ1322819_dateOfDeath",
    "date": {
      "label": "2010-05-08T00:00:00Z"

```

Figure 1: Results of a query service wrapping requests to LOBID (a) and WikiData (b) in IPIF

CA project and display them.⁵⁵

5 Conclusion and Future Work

This paper identified the need for a lightweight system promoting the reuse of prosopographical data and tools in different contexts. This system should be much simpler than the Web of Data proposals by the W3C and describe functionalities shared in individual prosopographical solutions. The paper therefore suggests a shared RESTful API. It describes the simple data model—an outcome of two workshops with developers and researchers working on prosopographical data—as well as the definition of the API. The API is built upon the well established “factoid” model which combines information on source, person, and statements extracted from the source in a factoid. Its publicly available definition following the OpenAPI standard allows automatic code generation.

As a proof of concept the paper has described how an existing biographical database (the Austrian Prosopographical information System - APIS) can easily provide its data via the API and how a new prosopographical database can be created using Python code generation tools and data extracted from the monasterium.net charter database. The common API consequently allows the reuse of a visualisation tool originally developed for the APIS database.

⁵⁵<https://github.com/GVogeler/prosopogrAPI/wiki/Tools:-Network-display-APIS>

This setup can be considered one prototypical use case for IPIF. We will continue the proof of concept with additional scenarios. We consider in particular data aggregation from different resources, simple autocomplete and look up activities, and the integration into information extraction pipelines in which IPIF can provide ground truth with labelled statementContent as interesting use cases for the API.

Further implementations should leverage RDF without taking the risk of maintenance of a SPARQL endpoint. We believe in taking the best of both worlds: the linked data capabilities and openness of RDF and the ease of use, maintainability and structure of RESTful APIs. For this we provide a preliminary JSON-LD context description.⁵⁶ If accepted by a larger community the API could allow scholarly editions, authority files, or biographical or family history databases to be reused in prosopographical information systems. First expressions of interest were made by the Personenregister des BBAW, scholarly editors at the Cologne Center for eHumanities,⁵⁷ and the Correspondence project.⁵⁸ In the end, a larger community experimenting with the definitions can demonstrate how the data model, the API definition including its compliance levels can contribute to reconstruct the social context of the people in the past.

⁵⁶<https://github.com/GVogeler/prosopogrAPI/blob/master/context.json>

⁵⁷From the Haller-project: <http://hallernet.org/>.

⁵⁸<http://correspsearch.net>

6 References

- Gerd Althoff. 1976. Zum Einsatz der elektronischen Datenverarbeitung in der historischen Personenforschung. *Freiburger Universitätsblätter*, 52:17–32.
- Suse Baeriswyl-Andresen. 2008. Das "Repertorium Academicum Germanicum": Überlegungen zu einer modellorientierten Datenbankstruktur und zur Aufbereitung prosopographischer Informationen der graduierten Gelehrten des Spätmittelalters. In *Städtische Gesellschaft und Kirche im Spätmittelalter. Arbeitstagung auf Schloss Dhaun 2004*, pages 17–36.
- John Bradley and Harold Short. 2005. Texts into Databases: The Evolving Field of New-style Prosopography. *Literary and Linguistic Computing*, 20(Suppl):3–24, January.
- Neithard Bulst. 1989. Prosopography and the computer: problems and possibilities. 2.
- Christophe Charle. 2015. Prosopography (Collective Biography). In James D. Wright, editor, *International Encyclopedia of the Social & Behavioral Sciences (Second Edition)*, pages 256–260. Elsevier, Oxford, January.
- Bernhard Ebneht and Matthias Reinert. 2017. Potentiale der Deutschen Biographie als historischbiographisches Informationssystem. *Europa baut auf Biographien: Aspekte, Bausteine, Normen und Standards für eine europäische Biographik*, pages 283–295.
- Fredo Erxleben, Michael Günther, Markus Kröttsch, Julian Mendez, and Denny Vrandečić. 2014. Introducing Wikidata to the Linked Data Web. In Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig Knoblock, Denny Vrandečić, Paul Groth, Natasha Noy, Krzysztof Janowicz, and Carole Goble, editors, *The Semantic Web – ISWC 2014, Lecture Notes in Computer Science*, pages 50–65. Springer International Publishing.
- Antske Fokkens and Serge ter Braake. 2018. Connecting People Across Borders: a Repository for Biographical Data Models. In *Proceedings of the Second Conference on Biographical Data in a Digital World 2017. Linz, Austria, November 6-7, 2017.*, volume 2119, pages 83–92. CEUR Workshop Proceedings.
- Karen Fox. 2019. The Cultural Journeys of Dictionaries of National Biography. In *True Biographies of Nations?*, page 19. ANU Press.
- Koen Goudriaan. 1995. *Prosopography and computer: contributions of mediaevalists and modernists on the use of computer in historical research*, volume 2. Garant.
- Arthur E. Imhof. 1978. The computer in social history: Historical demography in Germany. *Computers and the Humanities*, pages 227–236.
- Katharine SB Keats-Rohan. 2007. *Prosopography approaches and applications: A Handbook*, volume 13. Occasional Publications UPR.
- K. Faith Lawrence and Gabriel Bodard. 2015. Prosopography is Greek for Facebook: The SNAP: DRGN Project. In *WebSci*, pages 44–1.
- Paolo Missier, Khalid Belhajjame, and James Cheney. 2013. The W3c PROV family of specifications for modelling provenance metadata. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 773–776. ACM.
- Gerald Neumann, Fabian Körner, Thorsten Roeder, and Niels-Oliver Walkowski. 2010. Personendaten-Repositoryum. In *Berlin-Brandenburgische Akademie der Wissenschaften. Jahrbuch 2010*.
- N Ockeloën, A Fokkens, and Ter S Braake. 2013. BiographyNet: Managing provenance at multiple levels and from different perspectives. *Proceedings of the 3rd . . .*
- Michele Pasin and John Bradley. 2015. Factoid-based prosopography and computer ontologies: towards an integrated approach. *Literary and Linguistic Computing*, 30(1):86–97, April.
- Daniel L. Schwartz. 2019. Syriac Persons, Events, and Relations: A Linked Open Factoid-based Prosopography.
- Jouni Tuominen, Eero Hyvonen, and Petri Leskinen. 2017. Bio CRM: A Data Model for Representing Biographical Data for Prosopographical Research. page 8.