Digital Studies / Le champ numérique

Belouin, Pascal, Shih-Pei Chen and Sean Wang. 2021. "Designing an API-Based Protocol for the Interoperability of Textual Resources." *Digital Studies/Le champ numérique* 11(1): 10, pp. 1–17. DOI: https://doi.org/10.16995/dscn.8103

Open Library of Humanities

Designing an API-Based Protocol for the Interoperability of Textual Resources

Pascal Belouin, Max Planck Institute for the History of Science, DE, pbelouin@mpiwg-berlin.mpg.de
Shih-Pei Chen, Max Planck Institute for the History of Science, DE, schen@mpiwg-berlin.mpg.de
Sean Wang, Max Planck Institute for the History of Science, DE, swang@mpiwg-berlin.mpg.de

Designing a protocol for the interoperability of digital textual resources—or, more simply, a "IIIF for texts"—remains a challenge, as such a protocol must cater to their vastly heterogenous formats, structures, languages, text encodings and metadata. There have been many attempts to propose a standard for textual resource interoperability, from the ubiquitous Text Encoding Initiative (TEI) format to more recent proposals like the Distributed Text Services (DTS) protocol. In this paper, we introduce our proposal called SHINE, which prioritizes instead the ease for software developers to represent and exchange textual resources and their associated metadata. We do so by combining a hierarchical model of textual structure with a flexible metadata scheme in SHINE, and we continue to define and develop it based on user-centered and iterative design principles. Therefore, we argue that SHINE is a protocol for textual interoperability that successfully balances flexibility of resource representation, consistency across resource representation, and overall simplicity of implementation.

Concevoir un protocole pour l'interopérabilité des ressources textuelles numériques – c'est-à-dire, un IIIF pour des textes – demeure un défi, puisqu'un tel protocole doit correspondre à leurs formats considérablement hétérogènes, ainsi qu'à leurs structures, langues, encodages textuels et métadonnées. Il existe déjà plusieurs tentatives de proposer des standards pour l'interopérabilité des ressources textuelles, tel que l'ubiquiste Text Encoding Initiative (TEI – Initiative d'encodage textuel) ou des propositions plus récentes comme le protocole de Distributed Text Services (DTS – Services de texte distribuées). Dans cet article, nous présenterons une proposition que nous appelons SHINE, qui priorise la facilité de la représentation et de l'échange des ressources textuelles et des métadonnées associées pour les développeurs de logiciel. Nous le ferons en combinant un modèle de structure textuelle hiérarchique avec un schéma de métadonnées flexible dans SHINE et nous le définirons et le développerons selon des principes axés sur l'utilisateur et selon des principes de conceptions itératifs. Par conséquent, nous avançons que SHINE est un protocole pour l'interopérabilité textuelle qui équilibre systématiquement la flexibilité de la représentation de ressources, ainsi que la simplicité globale de l'implémentation, pour toute représentation de ressources.

1. Introduction

Internet and the digitization of sources profoundly changed the research process in the humanities. Researchers in text-centric disciplines like history and literature now access primary and secondary sources from different providers online in various formats (such as web pages, PDF files, Microsoft Word documents, or TEI documents). While many could be downloaded and saved locally (or on the cloud), others are locked in online read-only platforms. This diversity in both document formats and access avenues (including differing licenses) creates complications for scholars interested in computational and other large-scale analyses across texts from multiple providers. For instance, digital research tools for textual analyses, such as tagging or entity recognition tools, must ingest texts usually by manually uploading documents in specific formats. Besides technical work to pre-process the texts, many sources are covered by licenses that might make such upload difficult. Thus, while many sources and tools are already digital, researchers cannot easily combine and work across them: the varieties of formats and access rights necessitate significant manual curation; furthermore, certain licenses (especially on sources from commercial providers) make more-than-read analyses legally impossible.

To improve interoperability between digital research tools and textual resources on the web, many have proposed different legal and technical solutions, as well as large infrastructural projects that address a combination of both. As we summarize elsewhere (Wang et al. 2019), with few exceptions these proposals tend to bridge the gap by (re-) centralizing select resources and tools in a contained research environment. Since acquiring text-mining or similar more-than-read licenses from commercial providers can be expensive, such proposals require strong financial backers and are difficult to implement across institutions. Their centralization also limit flexibility for researchers and make reproducibility more challenging.

We were inspired by the success of International Image Interoperability Framework (IIIF) as an interoperable protocol for image-based research (Babcock and Di Cresce 2019). Following a similar distributed (or decentralized) approach, we have designed and implemented a research infrastructure for textual exchange called Research Infrastructure for the Study of Eurasia (RISE; Wang et al. 2021). We designed RISE to address these aforementioned issues in a pragmatic manner: RISE combines a robust authentication and authorization mechanism and clearinghouse (which checks and delivers exchanges based on agreed-upon rules) into an implemented middleware, with a relational state transfer (REST)-like application programming interface (API) text exchange protocol (which we call SHINE) that represents textual resources and their metadata in a flexible manner. While RISE was initially conceived to address specific challenges related to working with commercial textual resources in Chinese studies (Wang et al. 2018), its technical design works with textual resources regardless of language. We believe SHINE can do for textual resources analogous to what IIIF already did for online image collections. Said more simply, SHINE aims to provide a universal standard to seamlessly exchange textual resources as IIIF did for visual resources.

In this article, we focus primarily on the issue of designing a protocol for textual exchange and interoperability. First, we survey past proposals and current attempts for designing such a protocol to highlight some of the key requirements. We only focus on attempts that cover all three aspects of textual exchange and interoperability: discoverability, referencing, and interoperability of the actual resource contents; therefore, protocols that focus on discoverability or metadata only (such as OAI-PMH) are excluded. In this sense, we follow existing scholarship on language resource architecture and processing in computer science that understands interoperability as "the capacity of programs, components, representations, [and] data structures to interact" with one another (Witt et al. 2009, 5). Then we describe in detail our attempt (which is SHINE) and the rationale behind our design, which focuses on providing a generic and flexible representation for both structure and metadata that would cover the heterogeneity of textual resources. Our focus on minimal encoding of heterogeneous textual structures follows what Witt et al. (2009, 9) characterized as an "interlingua philosophy on interoperability," which constructs a representation that is a generalized abstraction over individual representations, in order for analyses and comparisons to be made across different texts and tools in a pragmatic manner. As a result, SHINE differs from other attempts of interoperable standards that encodes text contents in great detail for narrower, more specialized purposes (cf., Volodina et al. 2018; De Jong et al. 2020). We provide some practical examples to show how various textual resources could be represented in SHINE. Finally, we critically assess the future of interoperable text-based digital research and, considering past failures (Dombrowski 2014), the role research infrastructures could play in the ecosystem.

2. Existing protocols for representing and exchanging textual resources

Various attempts have been made to facilitate the interoperable exchange of textual resources since as early as 1987 (which predates the World Wide Web). The Text Encoding Initiative (TEI) initiated that year was an early attempt and has since been used by many projects, especially in literary studies. TEI guidelines have evolved and increased significantly in complexity; they are currently at their fifth iteration (Text Encoding Initiative 2021). Although most research projects today do not implement TEI with this as their main purpose, an early goal of TEI's was to "provide a standard format for data interchange in humanities research" (Text Encoding Initiative 1988). As Burnard (2013) noted, in an alternate universe we might have a more expansive TEI called "Text Encoding for Interchange." However, this ambition has fallen away and TEI instead focuses primarily on in-text markup rather than as a standard protocol for interoperable textual exchange. Therefore, in practice, any research software that automatically ingests TEI documents will have to confront this non-interoperability (Schmidt 2014). Furthermore, since TEI's inception predates the concept of web services, such use cases were not considered in its native design.

TEI's development history shows that achieving extremely fine-grained semantic interoperability within every text is fundamentally impossible given the level of encoding (and the resources) it would require, nor is it necessary for many scholarly purposes. With regards to interoperability on the web, more recent solutions have been proposed to allow for discoverability, referencing, and interoperability of textual resources (either in the TEI format or in plain text) that aims for some level of interoperability that is practical and corresponds to specific research purposes. The most historically significant effort in this direction is the Canonical Text Services protocol (CTS). Originally developed in 2010 as part of the Homer Multitext Project, CTS is a protocol designed to serve TEI texts (Blackwell and Smith 2014), and it allows well-written software clients to pull these texts from the Homer Multitext Project and, subsequently, from the Perseus Digital Library.

Digital historians who automatically consume texts from the Perseus Digital Library have identified lingering issues with CTS. For example, CTS relies on a Uniform Resource Name (URN) scheme, which can be slightly inconsistent across resources made available by the Perseus Digital Library (McPhee 2013). Clérice (2018) argued that one of CTS's main issues is the fact that it is "tightly coupled to text identifier syntax," which entails a number of drawbacks such as a lack of pagination for catalogs. Furthermore, CTS was initially developed by digital classicists as a project-specific standard, which makes wider adoption more challenging. Despite these shortcomings, CTS remains in use by Perseus and a few other projects such as Paraphrasis.org, but its uptake as a standard is limited.

A more recent development in the quest to create a protocol facilitating textual resource interoperability is the Distributed Text Services (DTS) initiative, which started in 2015. As DTS developers described it,

> The Distributed Text Services effort has been inspired, informed and influenced by the Canonical Text Services protocol (CTS). CTS has allowed many classical, canonical texts encoded in TEI to be made available in a machine-actionable, linked open

> data fashion. However, the CTS API is tightly coupled to the CTS URN identifier system which does not support citation systems used by more modern content or other forms of writing, such as papyri or inscriptions. The API also does not adhere to modern community standards for Web APIs (Distributed Text Services 2020).

In response, DTS adopts specifications of the Hydra Core Vocabulary (Lanthaler 2020), which combines the REST architectural style and Linked Data principles "to provide a vocabulary which enables a server to advertise valid state transitions." An API compatible with DTS provides three operation endpoints:

1. Collections Endpoint for navigating the text collection contents;
2. Navigation Endpoint for navigating within a single text document; and
3. Documents Endpoint to retrieve complete or partial texts.

The first two endpoints return JavaScript Object Notation for Linked Data (JSON-LD), a method to encode linked data using JSON, an open data exchange format for data consisting of attribute-value pairs and is in general the data format used for JavaScript and Web APIs, and the third returns the requested text or fragment primarily in TEI-XML. This results in a protocol that adheres strictly to the two most influential principles for modern web service design, REST and "hypermedia as the engine of application state" (HATEOAS; Fielding 2000), and arguably makes the DTS protocol future-proof, self-documented, and sufficiently flexible to represent any type of catalog of textual sources. Furthermore, its use of JSON-LD, a format increasingly popular among digital humanists, is another strong point. DTS's adherence to TEI's legacy, however, meant that alternative formats, including plain texts, always require additional calls and returns, which increase the required efforts for implementation. DTS also includes features for write-backs that alter texts on the server, as well as detailed citation-related designs in both document and navigation endpoints. Such features derived from TEI serve specialized purposes, yet are not necessary for simple exchanges of texts in the form of original or primary sources.

Our pursuit for simplicity and pragmatic implementation is why we developed separately another protocol for textual resource interoperability rather than adopting DTS. There are several reasons for this decision. The first is contextual. As we described elsewhere (Wang et al. 2018), we initially developed RISE as a distributed research infrastructure for Chinese studies and, as the necessity of an interoperable protocol became obvious, designed SHINE accordingly. This contextual legacy provided a certain degree of freedom; since we tackled the problem of textual representation and exchange from the ground up and focused in the first instance only on structured plain text, quite serendipitously we were able to avoid TEI's specter.

Our specific origin also resulted in important technical differences from DTS. Since RISE focuses on improving linkages between distributed textual collections and digital research tools, SHINE's initial design focused on this specific type of interoperability and, accordingly, targeted users who are primarily software developers rather than researchers. Our focus on the ease of implementation for developers (especially of digital research tools that consume textual resources) distinguishes SHINE from other attempts and results in more pragmatic design choices. In our opinion, software development practices usually develop organically and iteratively; they are informed by, but not fully adherent to, academic design and best practices. Indeed, the history of computer science is full of great ideas and concepts that were only marginally adopted by software development practitioners, such as the OSI model (Russell 2013) and the Semantic Web (Target 2018). Unfortunately, the strict constraints and principles from HATEOAS and pure RESTful web services are, at least for the time being, victims of the same fate. Practitioners have argued, for example, that "when designing a hypermedia API, you're really designing for a client that does not, and will never, exist" (Knupp 2014); one even went as far as calling HATEOAS "useless" (Reiser 2018). Cognizant of this history, we decided to design SHINE in a "REST-like" manner to better facilitate adoption and implementation by both resource providers and research tool developers.

In practice, this "REST-like" manner for developing APIs is already common. Most APIs by software development practitioners now consist of a set of routes that allows a client to perform "Create, Read, Update, and Delete" (or CRUD) operations on domain entities of a particular web service or application (see Open API Initiative 2018 for an example of this type of API commonly implemented for web services). Based on user-centered design principles (Gulliksen et al. 2003), SHINE includes a rather small number of entities that represent the structure of a textual resource, to which any type of metadata properties can be attached. These entities are then exposed by API routes which permit CRUD operations based on common industry guidelines (Hansson 2006). We also made a distinction between the act of "cataloging" textual resources and the simple representation of resources in terms of structure, content, and metadata.

In summary, although our goals are similar, SHINE and DTS represent resources and catalogs very differently in the API routes, and this difference stems primarily from development contexts. In designing SHINE, we have decided to sacrifice some strict adherence for a better balance between academic priorities and ease of uptake. SHINE and DTS remain complementary, however. As we design and develop SHINE in an agile, iterative manner, we have over time adopted DTS's many positive features, particularly its extensive use of JSON-LD to describe resources and the API scheme itself. One of us (Belouin) is also an observing member of the DTS Technical Committee, and there is an

API adaptor (in beta version) that can pull resources via DTS and make them available via SHINE endpoints (and vice versa). Given this complementarity, in the next section we describe SHINE in detail, focusing specifically on how it models and represents textual resources.

3. Overview of the SHINE protocol

Given our specific focus and from surveying existing attempts, we identified three criteria for evaluating a protocol for text interoperability: (1) flexibility of resource representation; (2) consistency across resource representation; and (3) overall simplicity of the protocol. These three criteria stand in tension, and in designing SHINE we have strived to achieve a balance among the three (with some trade-offs from each), rather than privileging one and sacrificing the others.

We also identified the following minimum aspects of textual resources that a protocol for interoperability must be able to model. First, a discrete textual resource (e.g., a book) is usually composed of hierarchical sub-parts down to the level of a single word; we refer to a resource's internal hierarchy as its "structure." Textual resource structures can vary greatly between genres (e.g., compare a newspaper article in 20th century England to a measured poem from the 7th century in Arabic), and a protocol must be flexible enough to represent these different structures. Second, a resource and its hierarchical sub-parts might have different properties or attributes, such as the language they are written in, their author, or other information relevant to a particular humanities discipline. One could also consider annotations made by a particular researcher about (a part of) a textual resource as an attribute. We refer to these types of information about a resource and its sub-parts "metadata." While some information might be considered as both structure and metadata (as seen in some TEI markups), we enforce clear distinction between these two types of information, as well as the "content" of a textual resource (i.e., plain texts), in SHINE.

Based on the above, we devised three main principles to guide the design of the SHINE protocol:

1. SHINE should be able to represent, with a limited number of entities, the structure of any genre of textual resources;
2. SHINE should be able to represent any type of metadata, whether associated with a textual resource or any of its sub-parts; and
3. SHINE should be simple enough so that any developer familiar with current industry standards in web services can implement it easily.

The SHINE protocol, in its current form, conforms to these principles. It is the result from an iterative design process where the resource representation model was continually refined as more resource providers and research tools developers adopted SHINE. We favor this pragmatic approach, as opposed to having researchers dictate design elements, in order to have SHINE closely reflect actual software development needs and encourage wider adoption. In other words, our approach takes after larger discussions regarding the design of distributed information architectures, especially with regards to RESTful web services (Witt et al. 2009, 9), and seeks to enable actually-existing workflows for both scholars and research software engineers in a pragmatic manner (cf., De Jong et al. 2020).

In a nutshell, the SHINE protocol consists of several REST-like routes that expose structural components of a textual resource. In addition, we also defined "collection" to group multiple resources. Thus, the building blocks used to represent structure in SHINE are—from the least to most granular—"collection," "resource," "section" (parts of a text, such as chapters or verses), and "content unit" (the smallest unit of text, which could be a page, a line, or even a word depending on the resource genre). Sections can have multiple hierarchical levels with a mechanism through which sections can be the children of another section, so that genres with more complex hierarchies (e.g., a novel with chapters, sub-chapter sections, and paragraphs) could still be represented as flat arrays using regular REST-like routes.

Table 1 shows the minimum set of API routes that a resource provider must implement to be SHINE-compatible. These seven routes allow any client to pull the structure, metadata, and content of the textual resources made available by a resource provider. Although the routes shown here are read-only, additional routes can also be implemented to allow POST, PUT, and DELETE requests. Thus, mapping this REST-like API structure to CRUD operations is straight-forward, and a resource provider can implement a software client to manage these additional operations.

HTTP Verb & Route	**Description**
GET/collections/	Returns all collections
GET/collections/*{uuid}*/	Returns the metadata for a collection
GET/collections/*{uuid}*/resources	Returns all resources for a collection
GET/resources/*{uuid}*/	Returns the metadata for a resource
GET/resources/*{uuid}*/sections	Returns the sections for a resource
GET/sections/*{uuid}*/	Returns the metadata for a section
GET/sections/*{uuid}*/content_units	Returns the content units for a section
GET/content_units/*{uuid}*/	Returns the metadata for a content unit

Table 1: The minimum set of API routes for a SHINE-compatible resource provider.

Each of these structural entities, or building blocks, can have metadata attached to them. The SHINE protocol represents, stores, and exchanges metadata as data objects that consist of attribute-value pairs and array data types, modeled after the JSON open standard (see **Figure 1**). Metadata fields are grouped by namespace, and we plan to make SHINE's metadata schema fully JSON-LD compliant in the next release.

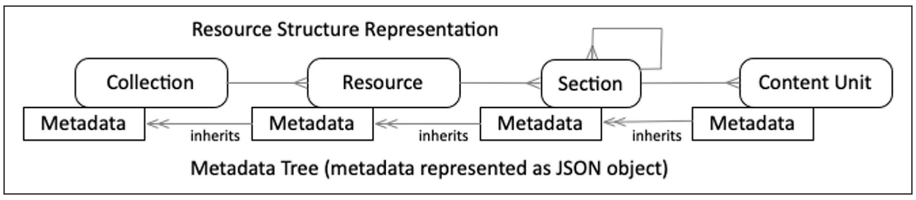

Figure 1: A schematic representation of the SHINE resource representation model.

Furthermore, as illustrated in **Figure 2**, these metadata objects are inherited from the higher structural entity in the hierarchy of a resource; however, a particular metadata field can be overridden at a lower structural entity. This metadata model with inheritance and override mechanisms provides flexibility, and we explore its full implication in the next section.

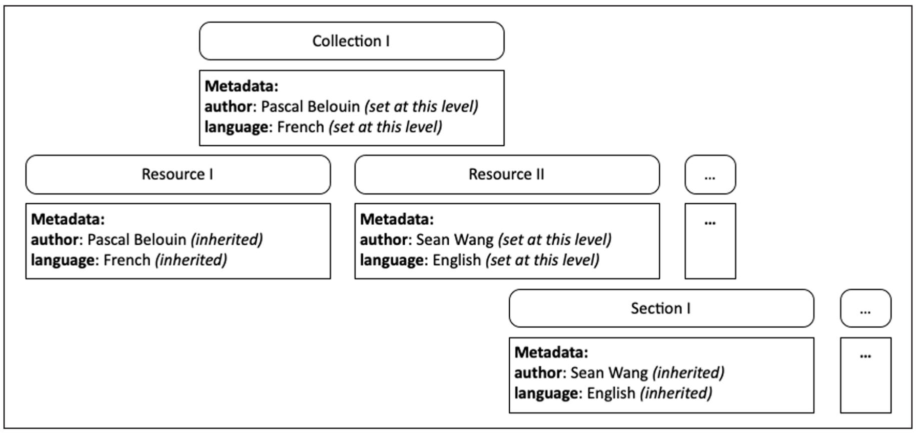
Collection I
Metadata:
author: Pascal Belouin (set at this level)
language: French (set at this level)
Resource I
Metadata:
author: Pascal Belouin (inherited)
language: French (inherited)
Resource II
Metadata:
author: Sean Wang (set at this level)
language: English (set at this level)
...
Section I
Metadata:
author: Sean Wang (inherited)
language: English (inherited)
...

Figure 2: A simplified example of SHINE's metadata inheritance and override mechanism.

3.1 Resource modeling

SHINE's modeling of textual resources is generic and flexible. It allows for any granularity for both structural and metadata representation. In general, SHINE's hierarchical structure allows for generic representation across genres, while the metadata inheritance and override mechanisms based on that hierarchy gives SHINE its flexibility. We demonstrate below several examples on how to work with this combination of structure and metadata to model multi-lingual resources in SHINE with ease.

Take a collection of Homer's classics, composed of various versions of *The Iliad* and *The Odyssey* (some of which were translated or commented on), as an example. **Figure 3** shows how this collection can be modeled in SHINE at higher levels of the structural hierarchy (collection and resources), while **Figure 4** shows the same collection at lower levels using one resource—the commented version of *The Odyssey*—with its sections and content units. We implement a basic set of Dublin Core fields in SHINE and fetch them from the resource providers, but any additional metadata information beyond that basic set—if provided by the resource providers—is still preserved. SHINE's metadata inheritance and override mechanisms work hierarchically alongside the resource structure. Any metadata information set at a higher structural level (e.g., collection) will be automatically inherited by lower levels. For example, once the creator (Homer) is set at the collection level, then all lower levels (resource, section, and content unit) in that collection will inherit the same metadata information. The same process is done with the language (Greek). However, one resource in this collection is an English translation of *The Iliad*. For this particular resource, we can override the inherited language metadata from the collection level (Greek) by changing it to English at the resource level. This override will not impact any higher level, but all lower levels of this resource (sections and content units) will inherit the overridden language metadata of English.

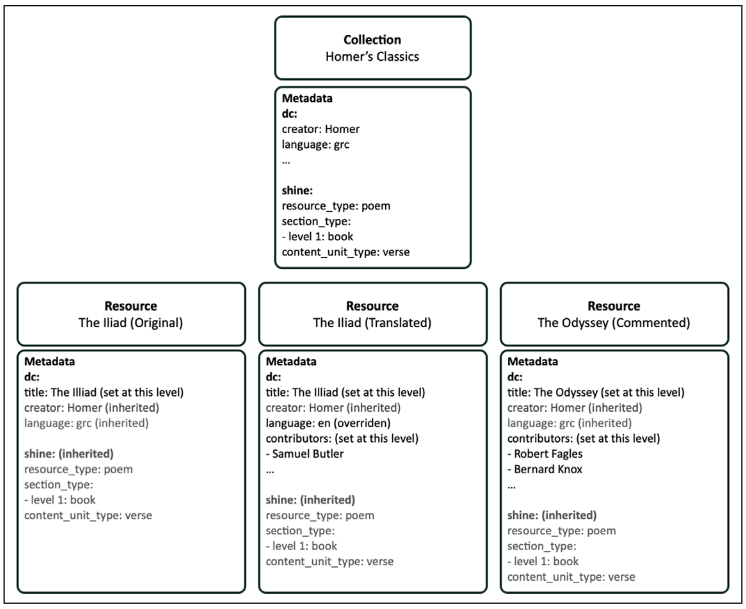

Figure 3: Representing a Homer's classics collection and its resources using SHINE.

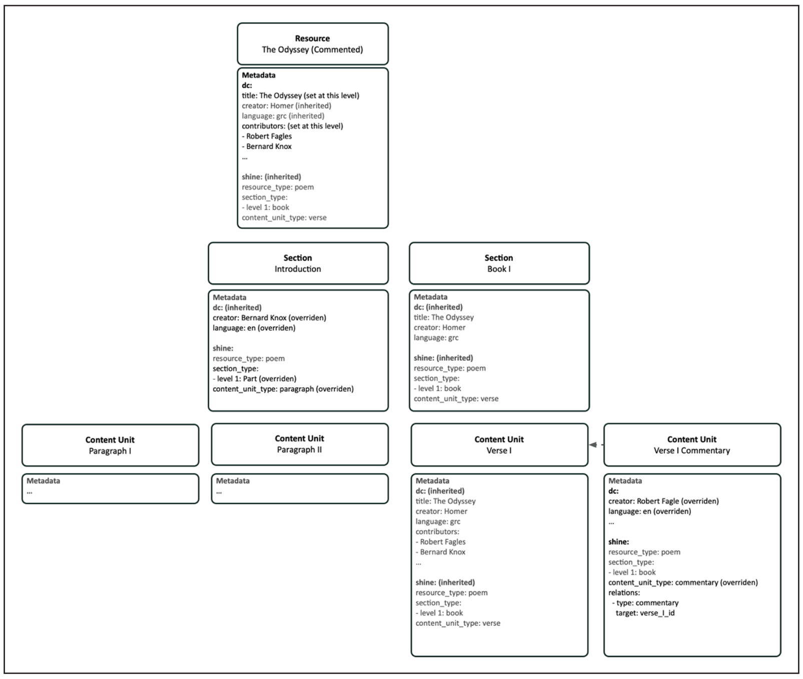

Figure 4: Representing various sections, content units, and their metadata using SHINE.

SHINE's metadata can also be used to store and represent information about how a particular structural entity maps onto specific genres (e.g., poems, paragraphs). This information can be set under the "shine" namespace of the metadata object. **Figure 4** shows an example using *The Odyssey*, where "section type" and "content unit type" are set. Since the structural entity "section" can have multiple hierarchical levels in SHINE (i.e., a section could have a parent section and multiple children sections), in some cases it is necessary to describe the type of section level in this metadata namespace. The SHINE metadata scheme could also represent a particular relationship between two objects of our model. If, for instance, a content unit contains a commentary or is a translation of another content unit, the "relations" metadata namespace can be used to 'point' to another content unit and to indicate the nature of this relationship. An example of this can be seen in **Figure 4** in the lower right corner.

In SHINE, metadata can either be overridden or added to at any level of the hierarchy. However, one disadvantage of this approach is the fact that one might need to access the metadata of a parent of a particular object (for instance a section's resource) to gain access to all of its metadata. We do hope that the implementation of helpers on the client side can alleviate this disadvantage of our design to some degree, allowing tool developers to integrate more easily with the RISE ecosystem.

3.2 Filtering and search

Filtering of resources becomes instinctively straight-forward given the REST-like routes provided by the SHINE protocol. Filtering based on a particular metadata field, for example, can be easily done by passing a particular filter string as a parameter to one of the collection routes, allowing for the filtering of collections, resources, or sections. Furthermore, we are currently working on defining a set of additional API endpoints and mechanisms to facilitate more advanced search based on metadata attributes, as well as full-text search where allowed.

3.3 Cataloging

Cataloging at the collection level could either be seen as a structural issue or a filtering based on some metadata information issue. We see many resource providers organize their resource catalog with the former approach. If we continue with the Homer's classics example above, a resource provider could create a collection called "Homer's classics," catalog various resources into this collection, and organize them in hierarchical trees. Based on the metadata information attached to them, resources could be grouped by theme, language, edition, etc., within this collection. DTS, for instance, employs this sophisticated method of cataloging that makes top-level collection a first-class citizen.

SHINE, on the other hand, employs the latter method and thus only permits one level of collection. We decided to sacrifice some theoretical sophistication for the sake of simple implementation and integration with authorization mechanisms on the collection routes. Structurally speaking, it is not possible to have a collection of collections in SHINE. Thus, grouping collections in SHINE must be done by filtering some metadata attributes, rather than using collections as a complex structural entity. For example, imagine that in addition to Homer's classics, two other collections are represented in SHINE: Jane Austen's complete works and *La Comédie humaine*. The latter consists of plain texts made available by Project Gutenberg, while the other two were published as collections by Penguin. It is not possible to have a collection of all collections published by Penguin on a structural level, but it is possible to do this *ad hoc* by filtering based on the metadata attribute "publisher: Penguin" on all collections.

3.4 Expanding the metadata object

Since the metadata object in SHINE is—in theory—infinitely expandable, we can use it for any type of description of a resource or its sub-parts. Thus, not only can metadata be used to store more conventional metadata information, but it could also include non-standard information beyond the plain text version of content (such as annotations and research data). Recall that in SHINE, we strictly enforce the distinction between structure and metadata in our resource modeling. This distinction works well with interoperability of plain texts but not so much for non-standard and in-text markups like in TEI. For those who are interested, we can utilize the full flexibility of SHINE's metadata scheme to do these TEI-like actions. Therefore, it is possible to, say, have one resource reference another, or parts of a resource to reference another part, internally or externally, in SHINE. This would allow us to model entities such as commentaries, as seen in **Figure 4**, where a relations namespace is used in the lower right corner.

3.5 Access rights management

Textual resources are covered by different licenses and resource providers might impose additional restrictions on their access rights. Therefore, in some cases resource access must be moderated. An authentication and authorization mechanism can be implemented relatively easily on top of SHINE's REST-like routes at any level of structural granularity. Indeed, we implemented access rights management in our own hosted middleware instance of RISE. One issue we have not resolved, however, is that more-than-read analyses often require transfer of texts. If the texts have very stringent restrictions on sharing, in the long-term we may have to implement some sort of encryption mechanism to fully comply with these kinds of restrictions.

4. Conclusion

The SHINE protocol described in this paper is a second version, and we are implementing it in the various infrastructural components of RISE. RISE currently links 130,772 resources to an increasing number of research tools. Besides plain texts, we are also working on mappers that would allow interoperability and transfer of TEI (and other XML-based textual data) through HTTP content negotiation. In this design and implementation process, we have solicited feedback from many stakeholders who work with digital texts and research infrastructures. As we advocate for a user-centered design, additional feedback from the community is always appreciated to help us refine our technical products iteratively. At the moment, we are especially looking for library partners. We believe that involvement and investment from additional stakeholders will approach our admittedly lofty ambition for a "IIIF for texts."

Although not the primary focus of this technical paper, SHINE's development process also illustrates the non-technical aspects of interoperability or, as De Jong et al. (2020, 3410) called it, "interoperability at the organisational level." In Europe, for example, large research infrastructures like CLARIN provide support at this level, as do many national-level initiatives. We recognize that these research infrastructures have accomplished an enormous amount, though as their organizational size grows, so does their distance from researchers' everyday needs. Our survey consistently shows that groups such as DTS have similar goals and origins as us, though our designs and scopes of implementation might be different. We have made SHINE-DTS mappers, and in our ongoing work with RISE it is possible that—depending on community uptake—we eventually replace SHINE with DTS entirely. Regardless of what might happen, we hope that the approach proposed here will shape the design of existing and future protocols so that we can together realize a "IIIF for text" within the digital humanities community.

Competing Interests

The authors have no competing interests to declare.

Author Contributions

Authors are listed in alphabetical order. Author contributions, described using the *CASRAI CredIT typology*, are as follows:

Author names and initials:

Pascal Belouin: PB

Shih-Pei Chen: SC

Sean Wang: SW

The corresponding author is Sean Wang

Conceptualization: PB, SC

Investigation: PB

Methodology: PB, SW

Project administration: SW

Resources: SC, SW

Software: PB

Visualization: PB, SW

Writing – original draft: PB, SW

Writing – review & editing: SW

Editorial contributions

Section and Copy Editor:

Shahina Parvin, The Journal Incubator, University of Lethbridge, Canada

References

Babcock, Kelli, and Rachel Di Cresce. 2019. "Impact of International Image Interoperability Framework (IIIF) on Digital Repositories." In *New Top Technologies Every Librarian Needs to Know*, edited by Kenneth J. Varnum, 181–96. Chicago: ALA Neal-Schuman.

Blackwell, Christopher, and Neel Smith. 2014. "The Canonical Text Services protocol, version 5.0.rc.2." Accessed September 10, 2021. http://cite-architecture.github.io/cts_spec/.

Burnard, Lou. 2013. "The Evolution of the Text Encoding Initiative: From Research Project to Research Infrastructure." *Journal of the Text Encoding Initiative* 5. DOI: https://doi.org/10.4000/jtei.811

Clérice, Thibault. 2018. “From File Interoperability to Service Interoperability: The Distributed Text Services.” Paper presented at the annual conference of the Text Encoding Initiative, Tokyo. https://hal.archives-ouvertes.fr/hal-02196659/.

De Jong, Franciska, Bente Maegaard, Darja Fišer, Dieter van Uytvanck, and Andreas Witt. 2020. “Interoperability in an Infrastructure Enabling Multidisciplinary Research: The Case of CLARIN.” In *Proceedings of the 12th Language Resources and Evaluation Conference*, edited by Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, et al., 3406–13. Marseille: European Language Resources Association. Accessed June 29, 2021. https://www.aclweb.org/anthology/2020.lrec-1.417/.

Distributed Text Services. 2020. “Distributed Text Services (DTS).” Accessed November 20, 2020. https://distributed-text-services.github.io/specifications/.

Dombrowski, Quinn. 2014. “What Ever Happened to Project Bamboo?” *Literary and Linguistic Computing* 29 (3): 326–39. DOI: https://doi.org/10.1093/llc/fqu026

Fielding, Roy Thomas. 2000. “Architectural Styles and the Design of Network-Based Software Architectures.” PhD diss., University of California, Irvine. Accessed November 20, 2020. https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

Gulliksen, Jan, Bengt Göransson, Inger Boivie, Stefan Blomkvist, Jenny Persson, and Åsa Cajander. 2003. “Key Principles for User-Centred Systems Design.” *Behaviour & Information Technology* 22 (6): 397–409. DOI: https://doi.org/10.1080/01449290310001624329

Hansson, David Heinemeier. 2006. “Resources on Rails.” Keynote presented at RailsConf, Chicago. Accessed November 20, 2020. https://youtu.be/GFhoSMD6idk.

Knupp, Jeff. 2014. “Why I Hate HATEOAS.” *Everything I Know about Python...* (blog), June 3. Accessed November 20, 2020. https://jeffknupp.com/blog/2014/06/03/why-i-hate-hateoas/.

Lanthaler, Markus. 2020. “Hydra Core Vocabulary: A Vocabulary for Hypermedia-Driven Web APIs.” Accessed November 20, 2020. http://www.hydra-cg.com/spec/latest/core/.

McPhee, Scot. 2013. “How to Retrieve Ancient Text Data from Perseus.” *inlustre monumentum est* (blog), April 10. Accessed November 20, 2020. https://inlustre.net/2013/04/how-to-retrieve-ancient-text-data-from-perseus/.

Open API Initiative. 2018. OpenAPI Specification v3.0.2. Accessed November 20, 2020. https://spec.openapis.org/oas/v3.0.2.

Reiser, Andreas. 2018. “Why HATEOAS is Useless and What That Means for REST.” Accessed November 20, 2020. https://medium.com/@andreasreiser94/why-hateoas-is-useless-and-what-that-means-for-rest-a65194471bc8.

Russell, Andrew L. 2013. “OSI: The Internet That Wasn’t.” *IEEE Spectrum*, July 30. Accessed November 20, 2020. https://spectrum.ieee.org/tech-history/cyberspace/osi-the-internet-that-wasnt.

Schmidt, Desmond. 2014. “Towards an Interoperable Digital Scholarly Edition.” *Journal of the Text Encoding Initiative* 7. DOI: https://doi.org/10.4000/jtei.979

Target, Sinclair. 2018. “Whatever Happened to the Semantic Web?” *Two-Bit History* (blog), May 27. Accessed November 20, 2020. https://twobithistory.org/2018/05/27/semantic-web.html.

Text Encoding Initiative. 1988. “Design Principles for Text Encoding Guidelines.” Accessed September 10, 2021. https://tei-c.org/Vault/ED/edp01.htm#b2b1b3b3b3.

---. 2021. “P5 Guidelines.” Accessed November 20, 2020. https://tei-c.org/guidelines/p5/.

Volodina, Elena, Maarten Janssen, Therese Lindström Tiedemann, Nives Mikelic Preradovic, Silje Karin Ragnhildstveit, Kari Tenfjord, and Koenraad de Smedt. 2018. “Interoperability of Second Language Resources and Tools.” In *CLARIN Annual Conference 2018 Proceedings*, edited by Inguna Skadina and Maria Eskevich, 90–94. Pisa: CLARIN. https://gup.ub.gu.se/publication/275365 http://www.clarin.eu/sites/default/files/CLARIN2018_Session-4-4_Paper-28_Volodina-Janssen-Lindstr%C3%B6mTiedemann-Preradovic-Ragnhildstveit-Tenfjord-deSmedt.pdf.

Wang, Sean, Pascal Belouin, Hou Ieong Ho, and Shih-Pei Chen. 2019. “RISE and SHINE: A Modular and Decentralized Approach for Interoperability between Textual Collections and Digital Research Tools.” Paper presented at the annual Digital Humanities Conference, Utrecht. Accessed September 7, 2021. https://dev.clariah.nl/files/dh2019/boa/0607.html.

Wang, Sean, Pascal Belouin, Shih-Pei Chen, Brent Ho, and Dagmar Schafer. 2021. “Research Infrastructure for the Study of Eurasia” Accessed September 06, 2021. https://www.mpiwg-berlin.mpg.de/research/projects/rise-and-shine-research-infrastructure-study-eurasia

Wang, Sean, Pascal Belouin, Shih-Pei Chen, and Hou Ieong Ho. 2018. “Research Infrastructure for the Study of Eurasia (RISE): Towards a Flexible and Distributed Digital Infrastructure for Resource Access via Standardized APIs and Metadata.” Paper presented at the 9th International Conference of Digital Archives and Digital Humanities, Taipei. Accessed September 07, 2021. https://pure.mpg.de/pubman/faces/ViewItemOverviewPage.jsp?itemId=item_3033461.

Witt, Andreas, Ulrich Heid, Felix Sasaki, and Gilles Sérasset. 2009. “Multilingual Language Resources and Interoperability.” *Language Resources and Evaluation* 43: 1–14. DOI: https://doi.org/10.1007/s10579-009-9088-x