Distributed Text Services (DTS): a Community-built API to Publish and Consume Text Collections as Linked Data

Bridget Almas, Hugh Cayless, Thibault Clérice, Vincent Jolivet, Pietro Maria Liuzzo, Jonathan Robie, Matteo Romanello, Ian W. Scott

Journal of the Text Encoding Initiative
Rolling Issue | 2023

Distributed Text Services (DTS): A Community-Built API to Publish and Consume Text Collections as Linked Data

Bridget Almas, Hugh Cayless, Thibault Clérice, Vincent Jolivet, Pietro Maria Liuzzo, Jonathan Robie, Matteo Romanello and Ian Scott

Electronic version
URL: https://journals.openedition.org/jtei/4352
DOI: 10.4000/jtei.4352
ISSN: 2162-5603

Publisher
TEI Consortium

Electronic reference
Bridget Almas, Hugh Cayless, Thibault Clérice, Vincent Jolivet, Pietro Maria Liuzzo, Jonathan Robie, Matteo Romanello and Ian Scott, “Distributed Text Services (DTS): A Community-Built API to Publish and Consume Text Collections as Linked Data”, *Journal of the Text Encoding Initiative* [Online], Rolling Issue, Online since 13 January 2023, connection on 15 January 2023. URL: http://journals.openedition.org/jtei/4352 ; DOI: https://doi.org/10.4000/jtei.4352

Distributed Text Services (DTS): A Community-Built API to Publish and Consume Text Collections as Linked Data

Bridget Almas, Hugh Cayless, Thibault Clérice, Vincent Jolivet, Pietro Maria Liuzzo, Jonathan Robie, Matteo Romanello, and Ian Scott

ABSTRACT

This paper presents the Distributed Text Service (DTS) API Specification, a community-built effort to facilitate the publication and consumption of texts and their structures as Linked Data. DTS was designed to be as generic as possible, providing simple operations for navigating collections, navigating within a text, and retrieving textual content. While the DTS API uses JSON-LD as the serialization format for non-textual data (e.g., descriptive metadata), TEI XML was chosen as the minimum required format for textual data served by the API in order to guarantee the interoperability of data published by DTS-compliant repositories. This paper describes the DTS API specifications by means of real-world examples, discusses the key design choices that were made, and concludes by providing a list of existing repositories and libraries that support DTS.

INDEX

Keywords: API Specification, Interoperability, FAIR, Text Navigation

AUTHOR'S NOTES

The authors contributed equally to this work.

1. Introduction

1 Digital humanities projects and libraries publish digital collections of texts for diverse reasons. Regardless of the original intent, once published the textual data itself could reach new audiences and support new research. It can do this by participating in the semantic web as linked data (Berners-Lee 2006). The Distributed Text Services (DTS) specification[1] aims to enable and facilitate the publication and consumption of text collections as linked data (Almas et al. 2018; Clérice et al. 2018).

2 DTS is a community-driven initiative that defines a hypermedia-driven Web Application Programming Interface (API) for working with collections of text as machine-actionable linked data. The DTS specification does not dictate how collections should be organized, what type of persistent identifiers should be used to reference them, what ontologies to use for metadata, how the texts are structured, or how the API is to be implemented. Instead, it aims to be as generic as possible, providing simple operations for navigating collections, navigating within a text, and retrieving textual content. By defining a standard, easily adoptable specification for navigating and interacting with a text collection as machine-actionable data, DTS hopes to provide a standard way to share and reuse collections of textual data.

3 In this paper we review related work in the realm of APIs for texts. We discuss the origins of the DTS effort, requirements for the API, and the rationale behind some of the decisions made. We then provide a detailed description of the API itself and explore some of its initial implementations. We conclude with a discussion of the feedback received on the initial public draft, and implementations and possibilities for future directions.

2. Related Work

4 The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) (Van de Sompel et al. 2004) is perhaps the earliest attempt at enabling interoperability and exchange of digital collections. Although not designed specifically for exchanging text collections, it can be used to this end. However, the genericity of OAI-PMH's design meant that this standard lacked certain features that are highly desirable when dealing with textual data, for example the ability to retrieve individual portions of a text without downloading and processing the content of an entire (possibly large) collection.

5 Concerning text APIs specifically, some concrete solutions have been developed over the last decade to enable the exchange of structured texts over standard protocols. The Canonical Text Services[2] (CTS) (Smith 2009; Blackwell and Smith 2019) was the first of such protocols to define an API as well as an identifier syntax—CTS URNs—to retrieve electronic texts. Unfortunately, some technological factors hindered CTS from becoming a widely adopted standard for exchanging texts:

1. The fact that the CTS protocol does not specify a required or preferred text encoding format (e.g., TEI/XML) leads to the inability of client applications to consume CTS-compliant API endpoints in a secure and reliable way;
2. Its strong commitment to the specificities of canonical texts, which makes it unsuitable for non-canonical material (e.g., archival documents, papyri, inscriptions, or works of modern literature);
3. Aspects of the design of the API, which keep it from scaling to large repositories of texts;[3]
4. The development of the CTS standard itself, which has been driven more by the needs of individual research projects rather than a community of practitioners;

5. The tight coupling of the API to the identifier scheme to be used for the texts which it served. It requires that texts be identified by a CTS URN.[4]

6 The absence of a widely adopted API to exchange structured texts has also led to the proliferation of ad-hoc solutions, whose designs often bear striking similarities to that of CTS and DTS. These include, for example, the API developed for the Scholastic Commentaries and Texts Archive (SCTA) (Witt 2018), the API that exposes the textual data of the School of Salamanca project,[5] or the SHINE Open API specification (Wang et al. 2019; Ho et al. 2018; Wang et al. 2018).[6] It is worth noting that, unlike other APIs, including DTS, SHINE was developed with the support for licensed resources and their secured access as a key requirement.

7 There is one thing that all these APIs do, each with its own custom conventions, namely, enable the interoperability of digital text collections available on the Web. At the same time, by not agreeing on a common standard, they contribute to a fragmented landscape in which each collection needs a dedicated client in order to be viewed and explored. This point is exactly what motivated the community to work on DTS, and a major source of inspiration in this work was provided by the community strategy, philosophy, and design of the Image Interoperability Framework (IIIF).[7] In fact, DTS is similar to IIIF in several respects: it develops community-driven technical specifications; it strives for genericity; it aims to support a distributed network of data providers.

3. Origins of DTS

8 The development of DTS was motivated by a desire to extend the possibilities made tangible by CTS to a broader range of texts and disciplines than the subset of classical texts that could adhere to CTS's strict canonical citation system and URN identifier scheme. Before initiating the development of DTS as a new specification, we explored the possibility of extending CTS. However, as a closed specification it was not open to community collaboration and thus was not suited for the extensions needed to meet the core requirements. These were:

- adherence to best practices for RESTful APIs;
- openness for community collaboration;
- support for any identifier scheme for collections, texts, and passages of text as long as it can be expressed safely as a URL parameter;

- support for collections of collections;
- support for texts with multi-level citation hierarchies;
- support for texts with citation hierarchies that vary within the text.

9 These requirements were fleshed out in both use cases and user stories, and can be found in the organization's GitHub repository, in the issues lists, and wiki.[8] In development of the initial draft specification, two decisions in particular were difficult.

10 The first was which standard to use for expressing and documenting the API. We wanted to use a standard that would facilitate both our own adherence to best practices and user adoption through widely available tooling to create and consume the API. The two main standards considered were the Open API Specification (OAS)[9] and the Hydra Specification (Hydra).[10] OAS (formerly Swagger Specification) is an API description format for REST APIs. An OpenAPI document that conforms to the OpenAPI Specification is itself a JSON object, which may be represented either in JSON or YAML format. Hydra is a vocabulary to create hyper-media driven Web APIs. A document which conforms to the Hydra specification can be expressed in JSON-LD. Although there was more tooling available for OAS, particularly in the form of code generators, we ultimately decided upon Hydra for its more stringent support of hyper-media API best practices, and especially the use of JSON-LD for expression of linked data.[11]

11 The second issue was whether or not to require TEI XML[12] as an output format for the API endpoint that returns the textual data (this is the documents endpoint, which is described more fully in the following section). TEI is a widely used and highly flexible standard for text encoding in the humanities. However, many projects offer their texts in other formats in addition to or instead of TEI (for example, as plain text, PDF, or HTML), which are *de facto* impoverishments of the TEI source. A primary motivator for developing the API in the first place was to facilitate interoperability and sharing of machine-actionable textual data. Supporting multiple formats as primary for text retrieval would have been counterproductive to that goal, because consumers would have to change their client code for each implementation of the API. For this reason, we decided to make TEI the required output format for textual data served by the API, which also allows for promoting the sharing of the richest data representation (i.e., TEI XML). This requirement does not prevent

repositories from making other formats available *in addition* to TEI or, conversely, to store their data by using other formats and then use TEI as a mere output format. Link headers can be used in these cases to notify clients that other formats are available.

4. The Specifications

4.1 General Architecture

12 To fulfill the requirements, the general architecture of the DTS specifications revolves around three resource models (see figure 1):

- the *collection*, which holds bibliographic metadata and serves as a catalog;
- the *reference*, which holds identifiers and metadata for subsection of document;
- the *passage*, which holds textual content.

13 These resources are each served by an endpoint and are linked through the representation of shared meta-objects. For example, if `example.com/identifier1` represents a book, the collection resource attached to this identifier will contain metadata about the book (author, publication date, etc.) and the passage resource will contain its actual full text. If this text can be represented in segments, references will provide metadata and segment identifiers for them.

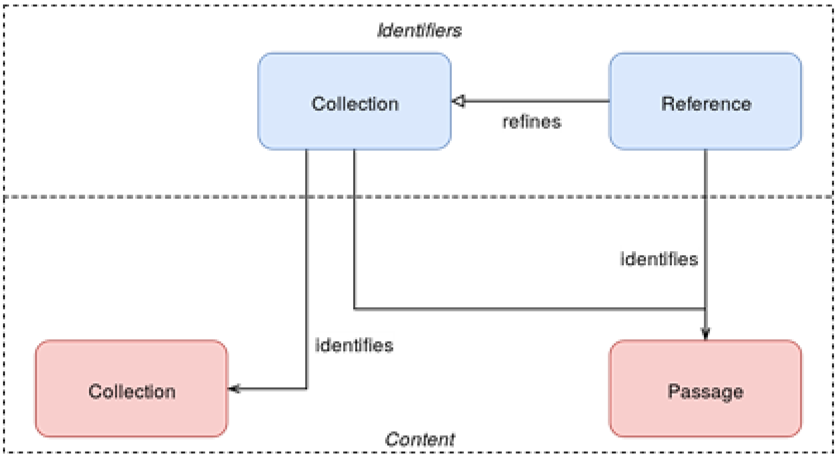
14 To take an example from the Alpheios Project's implementation of the DTS API,[13] if the Greek edition of Theogony by Hesiod is available as `urn:cts:greekLit:tlg0020.tlg001.alpheios-text-grc1`, the collection resource attached to this identifier (https://texts.alpheios.net/api/dts/collections?id=urn:cts:greekLit:tlg0020.tlg001.alpheios-text-grc1) will contain metadata about the work (author of the edition, title, etc.) and the passage resource (https://texts.alpheios.net/api/dts/document?id=urn:cts:greekLit:tlg0020.tlg001.alpheios-text-grc1) will contain its actual full text. If this text can be represented in passages, like chapters, sections, etc., then references (https://texts.alpheios.net/api/dts/navigation?id=urn:cts:greekLit:tlg0020.tlg001.alpheios-text-grc1) will provide metadata and references for them.

15 An example from the Beta maṣāḥəft Project[14] shows how a different sort of collection can be served in the same way: in the case of a manuscript transcription having the URI `https://betamasaheft.eu/BLorient718`, the collection resource attached

to this identifier (https://betamasaheft.eu/api/dts/collections?id=https://betamasaheft.eu/BLorient718)[15] will contain metadata about the manuscript (shelfmark, cataloguer, repository) and the passage resource (https://betamasaheft.eu/api/dts/document?id=https://betamasaheft.eu/BLorient718) will contain its actual full text. If this text can be represented in segments, like folia, columns, etc., then references (e.g., https://betamasaheft.eu/api/dts/navigation?id=https://betamasaheft.eu/BLorient718) will provide metadata and segment identifiers for them. In the case of manuscripts the smallest reference will be to lines, but in case additional referenceable structures are encoded, these will also become available as references.

16 The Epigraphic Database Heidelberg's implementation shows how the same collection model holds true if this is an inscription on stone, for example: https://edh-www.adw.uni-heidelberg.de/edh/inschrift/HD000001 (but note that this implementation does not currently support navigation).

Figure 1. Schema of the relations existing between Collection, Reference, and Passage, the three resource models defined by DTS.

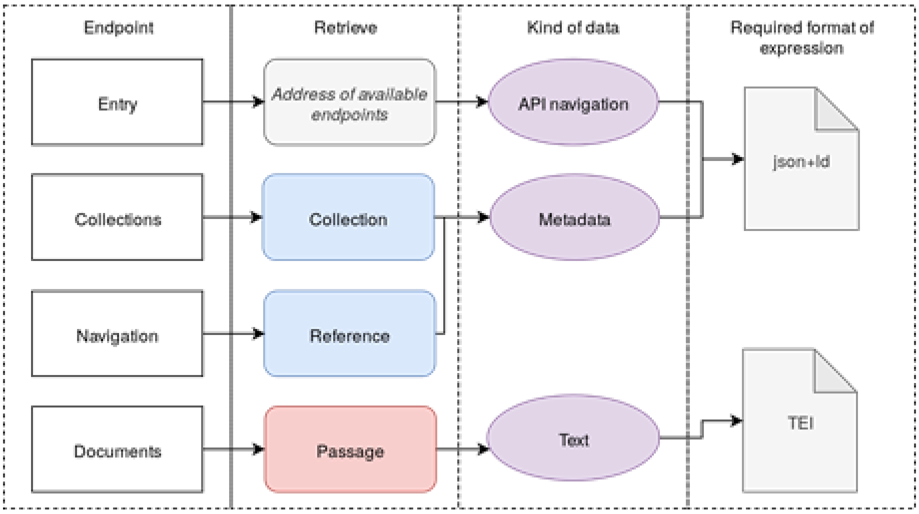


17 To link these three resource models and make them available, DTS provides specifications for three endpoints (or routes) that are respectively called Collections, Documents, and Navigation (see figure 2). To be DTS-compliant, a service need not implement all three endpoints. The Entry endpoint is the only mandatory one, as it declares and defines which other endpoints are implemented and are thus available. This was a design decision, made to support the use cases

of texts which cannot be served in passages, and of collections which describe texts but do not actually serve the contents. For example, it could be that a DTS API is itself an aggregator of catalogs, thus furnishing only a Collections endpoint.[16]

18 Throughout, the API reuses standard HTTP methods and URLs[17] for its interaction with the client: the GET method is used for reading content, URL query parameters identify resources and filter properties, and HTTP status codes are used as appropriate. The API uses a name-spaced vocabulary where necessary; for example, to provide links to other endpoints. This vocabulary is limited at the moment to predicates, and a minimalist approach is taken. We will now briefly outline each endpoint.

Figure 2. Schema representing the resource model underlying each DTS model, the type of data it exposes, as well as the required format of expression.



4.2 Collections Endpoint

19 The Collections endpoint is intended as a catalog-like entry point to the collections served by the API. It does not presuppose any particular organizational model for collections, thus leaving it entirely to the implementers to choose the catalog model that best suits their collections and needs. The catalog can be flat (where each collection at the root is readable), tree-based (where each collection can have only one parent), or graph-based (where each child can have multiple parents). Collections can be described as appropriate to the individual project or publisher, from something as stable as the FRBR model to ad-hoc approaches. A catalog could follow a typical

library organization, using authors and books as collections, or, such as in the context of an archaeological dig with inscriptions like Pompei, individual collections could be geographical areas in which the inscriptions were found.

20 The Collections endpoint makes use of just a few query parameters: `id`, `page`, and `nav`. The `id` parameter should contain the unique identifier of the resource and is reused throughout the API specifications. The parameters `page` and `nav` are browsing helpers, supporting pagination and hierarchical direction (i.e., traversing from parent to child or child to parent).

21 The Hydra vocabulary allows us to differentiate between two types of collections: `hydra:collection` and `hydra:resource`. A `hydra:resource` is a special type of `hydra:collection` that can be read (i.e., that represents or contains at least one Passage). Note though that a `hydra:resource` may contain children items too (hydra:collections and hydra:resources must have at least a title and can optionally have children and parents).

22 The model defines two metadata zones, differentiated by means of the properties `dts:dublincore` and `dts:extensions`. This separation aims to encourage the use of standard Dublin Core terms for metadata central to the description of collections, while still allowing for project-specific metadata and terminology in any namespace via the `dts:extensions` property. Example 1 shows how such an extension can be used.

Example 1. Request of the collection Carmagnole (identifier is https://www.wikidata.org/wiki/Q1043500) on the TNAH endpoint (http://tnah.chartes.psl.eu/2019/dts/collection): http://tnah.chartes.psl.eu/2019/dts/collection?id=https://www.wikidata.org/wiki/Q1043500.

```
{
  "@context": {
  "@vocab": "https://www.w3.org/ns/hydra/core#",
  "dc": "http://purl.org/dc/terms/",
  "dts": "https://w3id.org/dts/api#"
},
"@id": "https://www.wikidata.org/wiki/Q1043500",
"@type": "Resource",
"dts:citeDepth": 1,
"dts:dublincore": {
  "dc:creator": [
    "anonyme"
  ],
    "dc:date": [
      "1792"
    ]
  },
  "dts:extended": {
    "http://purl.org/ontology/mo/interpreter": [
      "anonyme"
    ]
  },
  "dts:passage": "/2019/dts/document?id=https%3A%2F%2Fwww.wikidata.org%2Fwiki
%2FQ1043500",
  "dts:references": "/2019/dts/navigation?id=https%3A%2F%2Fwww.wikidata.org%2Fwiki
%2FQ1043500",
  "title": "Carmagnole",
  "totalItems": 0
  }
```

4.2.1 Case Study 1—Different Recensions of a Text

23 It is common that texts have different versions even within one tradition. That is, the text was copied and altered with substantial changes. Although it remains in essence the same work, the alterations render it independent of other parallel recensions. For example, in the Ethiopic tradition, the *Physiologus* (Villa 2018, 145) has three such recensions, each with its own witnesses. In the Beta maṣāḥəft DTS Collection API, the SAWS[18] and CIDOC[19] ontologies are used in dts:extensions. In this example from https://betamasaheft.eu/api/dts/collections?id=https://betamasaheft.eu/LIT1401Physio, the general textual unit will contain references to the versions.

```
"dts:extensions" : {
  "saws:hasVersion" : [
    "https://betamasaheft.eu/LIT4916PhysB",
    "https://betamasaheft.eu/LIT4915PhysA",
    "https://betamasaheft.eu/LIT4917PhysC" ],
  "crm:P102_has_title" : [
    "https://betamasaheft.eu/LIT1401Physio/title/t3greek
    "https://betamasaheft.eu/LIT1401Physio/title/t1" ]
},
```

And, as in this example from https://betamasaheft.eu/api/dts/collections?id=https://betamasaheft.eu/LIT4915PhysA, each version will contain a statement to link it back to this general record, as well as a list of witnesses. Following the suggestion made by Cayless and Romanello (2021), the witnesses are expressed within the `dts:dublincore` property by using external vocabularies such as FaBiO (the FRBR-aligned Bibliographic Ontology).

```
"dc:source" : [ {
  "fabio:isManifestationOf" : "https://betamasaheft.eu/BNFet146",
  "@type" : "lawd:AssembledWork",
  "@id" : "https://betamasaheft.eu/BNFet146"
}, {
  "fabio:isManifestationOf" : "https://betamasaheft.eu/BLorient818",
  "@type" : "lawd:AssembledWork",
  "@id" : "https://betamasaheft.eu/BLorient818"
}, {
  "fabio:isManifestationOf" : "https://betamasaheft.eu/ONBAeth4",
  "@type" : "lawd:AssembledWork",
  "@id" : "https://betamasaheft.eu/ONBAeth4"
} ]
```

4.3 Navigation Endpoint

24 The Navigation endpoint answers the need to provide an index of available passages for a text when possible. It enables a client to provide browse functionality and also supports complex hierarchies in which each browsable passage may contain distinct passages. These passage resources, which the DTS specification names "References," can be grouped together in ranges (facilitating, for example, the use case of a text which might be segmented into thousands, or millions, of distinct passages which, for the purpose of browsing, are better off grouped in predefined chunks).

25 References are identified through a combination of the text identifier with the value of the `ref` query parameter. The `start` and `end` query parameters must be used to identify a range of passages and replaces the `ref` parameter. References can have their own metadata, reusing metadata constructs from the Collections endpoint (`dts:dublincore` and `dts:extensions`). This allows for descriptive metadata, which differs from the collection itself, such as is often the case with charters, cartularies, epistolary exchanges, journals, etc.

26 Supporting the identification and retrieval of passage ranges can be difficult for implementations to support. This might be due to the complexity of their texts (deep hierarchy, multiple structures) or other particulars of the data sources and software stack.[20] For this reason, the technical committee is working on adding a new property object to the specification that could be used to define implementations' capacities to deliver such functions.[21]

Example 2. On the TNAH endpoint, request for the references between one and ten of the Carmagnole: http://tnah.chartes.psl.eu/2019/dts/navigation?id=https%3A%2F%2Fwww.wikidata.org%2Fwiki%2FQ1043500&start=1&end=10&level=0.

```
{
  "@context": {
    "@vocab": "https://www.w3.org/ns/hydra/core#",
    "dc": "http://purl.org/dc/terms/",
    "dts": "https://w3id.org/dts/api#"
  },
  "@id": "/navigation?id=https%3A%2F%2Fwww.wikidata.org%2Fwiki
%2FQ1043500&start=1&end=10&level=0",
  "dts:citeDepth": 1,
  "dts:level": 1,
  "dts:passage": "/2019/dts/document?id=https%3A%2F%2Fwww.wikidata.org%2Fwiki
%2FQ1043500{&ref}{&start}{&end}",
  "member": [
    { "ref": 1  },
    { "ref": 2  },
    { "ref": 3  },
    { "ref": 4  },
    { "ref": 5  },
    { "ref": 6  },
    { "ref": 7  },
    { "ref": 8  },
    { "ref": 9  },
    { "ref": 10 }
  ]
}
```

4.3.1 Case Study 2—Excerpts Only

27 The text entitled “Malkəʾa Gabra Manfas Qəddus,” in the Beta maṣāḥəft DTS collection, has been identified in some manuscripts as being from *Dayr as-Suryān* in Egypt.[22] But in the process of cataloguing the manuscripts, few lines from the text, useful to identify it, have been copied, from the first and fourth stanza. Also, while adding this information the cataloguer added a label for only the first stanza, not for both those from which some text was copied. The Navigation endpoint at the time of that stage in the encoding reflected this situation by giving only the available information, but as soon as new stanzas or new labels will be added, these will be recorded as well.

```
{
  "dts:citeType": "stanza",
  "dts:level": 1,
  "@context": {
    "dts": "https://w3id.org/dts/api#",
    "@vocab": "https://www.w3.org/ns/hydra/core#",
    "dc": "http://purl.org/dc/terms/"
  },
  "@base": "/dts/api/document/",
  ...
  "member": [
    {
      "ref": "1",
      "dts:dublincore": {
        "dc:title": "Stanza 1"
      }
    },
    {
      "ref": "24"
    }
  ]
}
```

4.3.2 Case Study 3—Navigation of Texts with Different Levels of Hierarchy

28 Horace’s *Ars Poetica*[23] is a single poem, has a simple one-level navigation hierarchy, and one reference per line:

```
{
  ...
  "@id": "/navigation?groupBy=1&id=urn%3Acts%3AlatinLit%3Aphi0893.phi006.perseus-lat2&level=1",
  "dts:citeDepth": 1,
  "dts:level": 1,
  "dts:passage": "/documents?id=urn%3Acts%3AlatinLit%3Aphi0893.phi006.perseus-lat2{&ref}{&start}{&end}",
  "member": [
    { "ref": "1"},
    {"ref": "2"},
    {"ref": "3"},
    ...
    { "ref": "476"}
  ]
}
```

29 Whereas Horace's *Epistulae*,[24] a work composed of individual books and poems, has a three-level hierarchy. In this case the DTS API allows for fetching portions of the text at different hierarchical levels (book, poem, line in a poem):

```
{
  ...
  "@id": "/navigation?groupBy=1&id=urn%3Acts%3AlatinLit%3Aphi0893.phi005.perseus-lat2&level=1",
  "dts:citeDepth": 3,
  "dts:level": 1,
  "dts:passage": "/documents?id=urn%3Acts%3AlatinLit%3Aphi0893.phi005.perseus-lat2{&ref}{&start}{&end}",
  "member": [
    {
      "ref": "1"
    },
    {
      "ref": "2"
    }
  ]
}
{
  ...
  "@id": "/navigation?groupBy=1&id=urn%3Acts%3AlatinLit%3Aphi0893.phi005.perseus-lat2&level=2",
  "dts:citeDepth": 3,
  "dts:level": 2,
  "dts:passage": "/documents?id=urn%3Acts%3AlatinLit%3Aphi0893.phi005.perseus-lat2{&ref}{&start}{&end}",
  "member": [
    {
      "ref": "1.1"
    },
    {
      "ref": "1.2"
    },
    ...
    {
      "ref": "2.1"
    },
    {
      "ref": "2.2"
    }
  ]
}
{
  "@context": {
    "@vocab": "https://www.w3.org/ns/hydra/core#",
    "dc": "http://purl.org/dc/terms/",
    "dts": "https://w3id.org/dts/api#"
  },
  "@id": "/navigation?groupBy=1&id=urn%3Acts%3AlatinLit%3Aphi0893.phi005.perseus-lat2&level=3",
  "dts:citeDepth": 3,
  "dts:level": 3,
  "dts:passage": "/documents?id=urn%3Acts%3AlatinLit%3Aphi0893.phi005.perseus-lat2{&ref}{&start}{&end}",
  "member": [
    {
      "ref": "1.1.1"
    },
    {
      "ref": "1.1.2"
    },
    ...
    {
      "ref": "2.2.215"
    },
    {
      "ref": "2.2.216"
    }
  ]
}
```

4.4 Documents Endpoint

30 The Documents endpoint is intended for the retrieval of textual content as identified by a collection identifier, and optionally a reference. At a minimum, the endpoint must serve content in XML/TEI format, but if an implementation supports other output formats, these can be advertised via link headers and retrieved using HTTP content negotiation via request headers. XML/TEI was chosen as a standard and highly flexible model for representation of text (see section 3). However, it might be important for implementers to also offer plain text, HTML or ALTO XML, or other representations of requested passages.

31 A primary motivator for using DTS API is the sharing of machine-actionable textual content. The decision was made to limit the data returned by the Documents endpoint to the pure XML content of the text or passage. That means that this content is not modified, e.g., to add navigation links for next, previous, or parent passages. These links are provided in HTTP headers instead, using corresponding link relations.[25]

32 To support retrieval of fragments of text, the DTS API introduces under the DTS namespace an XML tag, `<dts:fragment>`, that can be used to identify the part where the requested fragment is found in the response, or can be used as a generic way to indicate that the resulting passage is a fabricated document.

Example 3. On the TNAH endpoint, request for the passage between reference 1 and 10 of the Carmagnole: http://tnah.chartes.psl.eu//2019/dts/document?id=https%3A%2F%2Fwww.wikidata.org%2Fwiki%2FQ1043500&start=1&end=10.

```
<TEI xmlns:dts="https://w3id.org/dts/api#">
 <teiHeader>
  <fileDesc>
   <titleStmt>
    <title>Carmagnole</title>
    <author>anonyme</author>
   </titleStmt>
  </fileDesc>
 </teiHeader>
 <dts:fragment>
  <text>
   <body>
    <l n="1">Madam’ Veto avait promis (bis)</l>
    <l n="2">De faire égorger tout Paris (bis)</l>
    <l n="3">Mais son coup a manqué</l>
    <l n="4">Grâce à nos canonniers.</l>
    <l n="5">Dansons la Carmagnole</l>
    <l n="6">Vive le son (bis)</l>
    <l n="7">Dansons la Carmagnole</l>
    <l n="8">Vive le son du canon !</l>
    <l n="9">Dansons la Carmagnole</l>
    <l n="10">Vive le son (bis)</l>
   </body>
  </text>
 </dts:fragment>
</TEI>
```

5. Existing Implementations and APIs

33 The DTS specification has already been implemented by a few projects in the scholarly community. We propose to divide them by corpus size:

- Small Corpora
    - Ecole Nationale des Chartes covers two periods for its corpora: contemporaneous and medieval. Contemporaneous are lightly marked up, while medieval are finely annotated.[26]

  - Alpheios.net provides a small collection of Latin and Greek texts which have been aligned with linguistic annotation for learning ancient languages.[27]
- Medium Sized Corpora (> 1000 texts)
  - Perseids serves all textual resources available from Perseus within the Ancient Greek and Latin corpora as well as some resources in Hebrew and Farsi.[28]
  - Beta maṣāḥəft collects written artefacts from the highlands of Ethiopia and Eritrea mainly in *Gəʿəz* (Classical Ethiopic).[29] In the collection are present both transcriptions of manuscripts and editions of textual units. The scarce availability of transcriptions as well as available editions means that the actual text contents are few in comparison with the textual units and written artefacts identified and described.
  - Epigraphische Datenbank Heidelberg holds around 80,000 short texts from Latin epigraphic databases.[30]

34 In the realm of shared tools, two client libraries for DTS have been implemented:

- the TEI Publisher client, which serves as a web interface for DTS APIs: it supports browsing collections and document retrieval, but not navigation (Turska 2019);
- The MyCapytain implementation from the capitains.org project, which is a Python implementation in the form of a library (Clérice et al. 2019).

35 MyCapytain, via Nautilus,[31] and the TEI Publisher both provide a server API, with the latter having the same limitations as the client (i.e., only Collection and Document endpoints are supported). Another implementation for small corpora is offered by the DTS-Demo-Server in Python with an SQL database.[32] It supports one level only in navigation (Dartois, Vieillon, and Clérice 2019).

6. Conclusions

36 Implementing a DTS-compliant API contributes to the efforts of publishers of text collections to adhere to both FAIR[33] (Boeckhout et al. 2018; Wilkinson et al. 2016) and 5-Star Linked Data principles for their textual data. In particular, DTS:

- encourages publishers to use stable persistent identifiers for their texts and their collections
- supports the use of standard vocabularies for the descriptive metadata and enables expression of that metadata separately from the textual content itself
- provides documented (but unconstrained) access to the information about the structure of a textual resource, down to the level of a citation
- enables detailed specifications of relations among resources
- has no canonical implications as relations between texts depend on the collection
- does not impose any requirements upon how data is stored
- is extensible, open, free, and universally implementable

37 One of the main tenets of the DTS specification effort since its inception has been the emphasis on community input and collaboration. Meeting minutes are posted openly in the organization’s GitHub repository[34] and a Google Groups discussion list, which as of this writing has forty-five members and is open to anyone to join. Submission of requests in the form of GitHub issues is encouraged and participation on the technical committee is welcome to those who make a commitment to active engagement. Additional roles of Ambassador and Contributor have been defined to enable diverse contributions.[35]

38 The first draft specification was released in autumn 2018 and has already received considerable community feedback. Among the top requests being considered for future implementation are:

- documented guidelines for integration with IIIF APIs that serve image resources (to facilitate linking of textual data with relevant images, such as those of the original manuscripts or inscriptions)
- documented guidelines for expressing versioning of textual resources
- endpoints for indexing and searching

39 One need which falls a bit outside of the scope of the DTS effort is for a centralized resolution service that could locate a text or texts from within a network of distributed DTS implementations (Cayless and Romanello 2021). Making this a reality will require community support for shared services (e.g., a catalog of known DTS APIs) and agreement on vocabularies and best practices for metadata.

40 Notwithstanding the work that remains to be done, DTS can already facilitate the sharing and reuse of textual data. Community interest and bandwidth will determine the next steps, and the speed of progress, towards a fully open, interoperable ecosystem of texts as linked, machine-actionable data.

BIBLIOGRAPHY

Almas, Bridget, Thibault Clérice, Jonathan Robie, Hugh A. Cayless, Zacary Fletcher, Vincent Jolivet, Pietro Maria Liuzzo, et al. 2018. “Distributed Text Services API Specification.” Accessed October 29, 2019. https://w3id.org/dts.

Almas, Bridget, and Caroline Schroeder. 2016. “Applying the Canonical Text Services Model to the Coptic SCRIPTORIUM.” *Data Science Journal* 15. doi:10.5334/dsj-2016-013.

Berners-Lee, Tim. 2006. “Linked Data - Design Issues.” July, 2006. http://www.w3.org/DesignIssues/LinkedData.

Blackwell, Christopher W., and Neel Smith. 2019. “The CITE Architecture: A Conceptual and Practical Overview.” In *Digital Classical Philology: Ancient Greek and Latin in the Digital Revolution*, edited by Monica Berti, 73–93. Berlin, Boston: De Gruyter Saur. doi:10.1515/9783110599572-006.

Boeckhout, Martin, Gerhard A. Zielhuis, and Annelien L. Bredenoord. 2018. “The FAIR Guiding Principles for Data Stewardship: Fair Enough?” *European Journal of Human Genetics* 26: 931–936. doi:10.1038/s41431-018-0160-0.

Cayless, Hugh A. 2019. “Sustaining Linked Ancient World Data.” In *Digital Classical Philology: Ancient Greek and Latin in the Digital Revolution*, edited by Monica Berti, 35–50. Berlin, Boston: De Gruyter Saur. doi:10.1515/9783110599572-004.

Cayless, Hugh, and Matteo Romanello. 2021. “Towards Resolution Services for Text URIs.” In *Graph Data-Models and Semantic Web Technologies in Scholarly Digital Editing*, edited by Elena Spadini, Francesca Tomasi, and Georg Vogeler, 31–44. Norderstedt: Books on Demand. Accessed October 29, 2019. https://kups.ub.uni-koeln.de/55223/.

Clérice, Thibault, Bridget Almas, Hugh Cayless, Vincent Jolivet, Emmanuelle Morlock, Jonathan Robie, James Tauber, Jeffrey Witt, and Pietro Maria Liuzzo. 2018. "From File Interoperability to Service Interoperability : The Distributed Text Services." TEI 2018, September 2018, Tokyo, Japan. Accessed October 29, 2019. https://hal.archives-ouvertes.fr/hal-02196659.

Clérice, Thibault, Matthew Munson, and Bridget Almas. 2019. "Capitains/MyCapytain: 3.0.0." Zenodo. doi:10.5281/zenodo.3490954.

Dartois, Hélène, Lucie Vieillon, and Thibault Clérice. "Chartes-TNAH/Dts-Demo-Server: V1.0.0." Zenodo. doi:10.5281/zenodo.3522043.

Ho, Brent, Sean Wang, Pascal Belouin, and Shih-Pei Chen. 2018. "Asia Network : An API-Based Cyberinfrastructure for the Flexible Topologies of Digital Humanities Research in Sinology." In *Pacific Neighborhood Consortium Annual Conference and Joint Meetings (PNC)*, 1–4. doi:10.23919/PNC.2018.8579459.

Smith, Neel. 2009. "Citation in Classical Studies." *Digital Humanities Quarterly* 3 (1). Accessed October 29, 2019. http://www.digitalhumanities.org/dhq/vol/003/1/000028.html.

Tiepmar, Jochen, Christoph Teichmann, Gerhard Heyer, Monica Berti, and Gregory Crane. 2014. "A New Implementation for Canonical Text Services." In *Proceedings of the 8th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*, 1–8. Gothenburg, Sweden: Association for Computational Linguistics. doi:10.3115/v1/W14-0601.

Turska, Magdalena. "TEI Publisher 5.0.0." *TEI Publisher* (blog), August 2, 2019. Accessed October 29, 2019. https://teipublisher.com/exist/apps/tei-publisher/doc/blog/tei-publisher-50.xml.

Van de Sompel, Herbert, Michael L. Nelson, Carl Lagoze, and Simeon Warner. 2004. "Resource Harvesting within the OAI-PMH Framework." *D-Lib Magazine* 10 (12. doi:10.1045/december2004-vandesompel.

Villa, Massimo. 2018. "Encoding Strategies and the Ethiopic Literary Heritage: The *Physiologus* as a Case Study." *COMSt Bulletin* 4 1: 143–49.

Wang, Sean, Pascal Belouin, Shih-Pei Chen, and Brent 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." In *DADH. 9th International Conference of Digital Archives and Digital Humanities*, 21–37. Taipei: Dharma Drum Institute of Liberal Arts. doi:21.11116/0000-0003-2EFA-1.

Wang, Sean, Belouin Pascal, Hou Ieong Ho, and Chen Shih-Pei. 2019. "RISE and SHINE: A Modular and Decentralized Approach for Interoperability between Textual Collections and Digital Research Tools." Digital Humanities Conference 2019, 9–12 July 2019. doi:https://doi.org/10.34894/9BF3NR.

Wilkinson, Mark D., Michel Dumontier, Isbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, et al. 2016. "The FAIR Guiding Principles for Scientific Data Management and Stewardship." *Scientific Data* 3. doi:10.1038/sdata.2016.18.

Witt, Jeffrey C. 2018. "Digital Scholarly Editions and API Consuming Applications." In *Digital Scholarly Editions as Interfaces*, edited by Roman Bleier, Martina Bürgermeister, Helmut W. Klug, Frederike Neuber, and Gerlinde Schneider, 219–47. Schriften des Instituts für Dokumentologie und Editorik 12. Norderstedt: Books on Demand. Accessed October 29, 2019. https://kups.ub.uni-koeln.de/9118/.

NOTES

[1] Accessed October 29, 2019, https://www.w3id.org/dts.

[2] Accessed October 29, 2019, https://web.archive.org/web/20200201014458/http://cite-architecture.org/cts/.

[3] In terms of scalability, the most problematic aspect of the design of CTS is certainly the `GetCapabilities` method, which returns the full catalog of documents available in a given CTS endpoint. Since this method, by design, does not make use of scalability-friendly solutions (such as the pagination of the API response), any CTS implementation will be unable to scale to collections containing hundreds of thousands of texts, independently from the length of each individual text, due to the design limitations of the `GetCapabilities` method itself. While Tiepmar et al. (2014) investigated the issue of the scalability of CTS implementations, they considered a maximum of 10,000 editions in a CTS endpoint for their benchmarking experiments, whereas the number of texts contained in the largest known DTS implementation is seven times bigger (Epigraphische Datenbank Heidelberg, see section 5.

[4] On the complexities of applying the CTS URN scheme to an existing collection see Almas and Schroeder (2016).

[5] Accessed October 29, 2019, https://github.com/digicademy/svsal/blob/master/docs/API.md.

[6] Accessed October 29, 2019, https://rise.mpiwg-berlin.mpg.de/.

[7] Accessed October 29, 2019, https://iiif.io/.

[8] Accessed October 29, 2019, https://github.com/distributed-text-services/specifications.

[9] Accessed October 29, 2019, https://swagger.io/specification/.

[10] Accessed October 29, 2019, https://www.hydra-cg.com/spec/latest/core/.

[11] On the simplicity of JSON-LD as one of the key success factors of linked data systems, see (Cayless 2019).

[12] Accessed October 29, 2019, https://tei-c.org/.

13 Accessed October 29, 2019, https://texts.alpheios.net/api/dts.

14 Accessed October 29, 2019, https://betamasaheft.eu.

15 The current `urn:dts:` ids are temporary and will be replaced with standard URIs soon, without this affecting the example or the navigability of the API.

16 This also means that projects which are in constant change and frequently updated live, do not have to wait and can provide what is available. Sometimes, for example, a project may have passages and references, but no text, so that it is still possible to know that there is a passage although no text is available for it.

17 Accessed October 29, 2019, https://www.w3.org/Protocols/rfc2616/rfc2616.html.

18 Accessed October 29, 2019, http://www.ancientwisdoms.ac.uk/.

19 Accessed October 29, 2019, http://www.cidoc-crm.org/.

20 For example, if the implementers were using self-closing XML elements in XML source files for their application, range, and reference in general it can be quite difficult to implement if the stack does not include XPath 2.

21 See https://github.com/distributed-text-services/specifications/issues/164 and the minutes of the meeting https://github.com/distributed-text-services/meeting-notes/blob/master/notes/2019-10-11.md.

22 See https://betamasaheft.eu/DSintro.html. Example from https://betamasaheft.eu/api/dts/navigation?id=https://betamasaheft.eu/LIT5068MalkeaGabraManfas, accessed October 29, 2019.

23 In the Perseids DTS API at: https://dts.perseids.org/navigation?id=urn:cts:latinLit:phi0893.phi006.perseus-lat2, accessed October 29, 2019.

24 In the Perseids DTS API at: https://dts.perseids.org/navigation?id=urn:cts:latinLit:phi0893.phi005.perseus-lat2, accessed October 29, 2019.

25 The `next` and `prev` link relations are registered with IANA (https://www.iana.org/assignments/link-relations/link-relations.xhtml), and defined in https://html.spec.whatwg.org/multipage/links.html#sequential-link-types.

26 Accessed October 29, 2019, https://dev.chartes.psl.eu/api/nautilus/dts.

27 Accessed October 29, 2019, https://texts.alpheios.net/api/dts.

28 Accessed October 29, 2019, https://dts.perseids.org/.

29 Accessed October 29, 2019, https://betamasaheft.eu/api/dts.

30 Accessed October 29, 2019, https://edh-www.adw.uni-heidelberg.de/api/dts/.

31 Accessed October 29, 2019, https://github.com/Capitains/Nautilus.

32 Accessed October 29, 2019, https://github.com/Chartes-TNAH/dts-demo-server.

33 Accessed October 29, 2019, https://www.force11.org/group/fairgroup/fairprinciples.

34 Accessed October 29, 2019, https://github.com/distributed-text-services/meeting-notes.

35 Accessed October 29, 2019, https://distributed-text-services.github.io/specifications/Organization.html.

AUTHORS

BRIDGET ALMAS

Bridget Almas is Director of Data Innovation Strategy at the State University of New York. She has over twenty-five years experience working in software development, including commercial, academic, and non-profit environments. In prior roles she was Executive Director and Software Architect for The Alpheios Project, building evidence-based, open-source software to support the worldwide study of classical languages and literatures. Before that, Bridget was the technical lead on both the Perseids Project and the Perseus Digital Library. She has acted in several leadership roles in the Research Data Alliance, and as a liaison between the Alliance of Digital Humanities Organizations and the Research Data Alliance.

HUGH CAYLESS

Hugh Cayless is the Senior Digital Humanities Developer at Duke University Libraries. He has over two decades of software engineering expertise in both academic and industrial settings. He holds a PhD in Classics and a Master's degree in Information Science. He is one of the founders of the EpiDoc collaborative and currently serves on the Technical Council of the Text Encoding Initiative. He also serves as the Treasurer of the TEI Consortium.

THIBAULT CLÉRICE

Thibault Clérice is a digital humanist with a classical studies background, who served as an engineer both at the Centre for eResearch (Kings College London, UK) and the Humboldt Chair for Digital Humanities (Leipzig, Germany) where he developed the data backbone of the future Perseus 5 (under the CapiTainS.org project). He was head of the DH applied to GLAM program for 5 years at the École nationale des Chartes. His research mainly focus on natural language processing for ancient languages through deep learning, the distribution of corpora and computational methods applied to the humanities.

VINCENT JOLIVET

Vincent Jolivet is head engineer at the École Nationale des Chartes and has been participating for several years in various research programs in Digital Humanities (ENC/PSL) at the Université Paris-Sorbonne. These programs are devoted to the production and automatic analysis of large textual corpora, the aim of which is to break down data silos and increase interoperability for texts from different eras and of different types.

PIETRO MARIA LIUZZO

Pietro Maria Liuzzo is Senior Information Scientist at the Fotothek of the Bibliotheca Hertziana, Max-Planck-Institut für Kunstgeschichte in Rome. He was until February 2022 in charge of the digital aspects of the project Beta maṣāḥəft: Manuscripts of Ethiopia and Eritrea (Schriftkultur des christlichen Äthiopiens und Eritreas: eine multimediale Forschungsumgebung). He deals with digital images, infrastructure architecture and applications for art history objects, written artefacts, manuscripts, and inscriptions, as well as their digital representations, encoding, visualization, and reusability. After completing his PhD in Ancient Greek Historiography on FGrHist 104 at Studiorum Università di Bologna, he worked at the University of Heidelberg for the EAGLE (European Network of Ancient Greek and Latin Epigraphy) project as networking coordinator.

JONATHAN ROBIE

Jonathan Robie is the Principle Engineer for Clear Bible's MACULA project, which includes linguistic datasets for biblical Hebrew and biblical Greek and a research environment for working with biblical texts. Previously, he was Program Manager for the Paratext ecosystem, used by over 9,000 Bible translators worldwide. He is also co-chair of the Copenhagen Alliance for Open Biblical Resources. Jonathan was a lead editor for the W3C XPath and XQuery specifications, chair of the API Governance Board at EMC's Enterprise Content Division, a member of the AMQP enterprise messaging team at Red Hat, and the architect of XML database systems at Software AG, Progress Software, Texcel Incorporated, and POET Software.

MATTEO ROMANELLO

Matteo Romanello is Ambizione SNF Lecturer at the University of Lausanne, where he conducts a project on the commentary tradition of Sophocles' Ajax. Matteo is a Classicist and a Digital Humanities specialist, with expertise in various areas of the Humanities, including archaeology and history. After obtaining his PhD from King's College in London, he worked as a research scientist at EPFL's DHLAB on the Linked Books and Impresso projects, before moving to his current position. He was also a teaching fellow at the University of Rostock, a researcher at the German Archaeological Institute, and a visiting research scholar at Tufts University.

IAN SCOTT

Ian Scott is Associate Professor of New Testament at Tyndale Seminary in Toronto, Canada. An historian of Second Temple Judaism and Early Christianity, he is founder and co-editor of the Online Critical Pseudepigrapha. This project encodes and publishes manuscripts and reconstructed texts of the so-called "Old Testament Pseudepigrapha" and related Jewish and Christian writings. As part of that work he has developed a variety of user interfaces for consuming ancient texts, and he is interested in developing standards to allow such tools to be more interoperable with a variety of textual sources.