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

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

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

Bridget Almas[a], Thibault Clérice[bc], Hugh Cayless[d], Vincent Jolivet[b], Pietro Maria Liuzzo[e], Matteo Romanello[f], Jonathan Robie[g], Ian W. Scott[h]

[a] *The Alpheios Project, Ltd.*
[b] *Centre Jean Mabillon, École nationale des chartes, PSL Research University*
[c] *Hisoma, Université de Lyon*
[d] *Duke University*
[e] *Hiob Ludolf Centre for Ethiopian Studies, Universität Hamburg*
[f] *Université de Lausanne*
[g] *SIL International*
[h] *Tyndale Seminary, Toronto*

Abstract. The paper presents the Distributed Text Service 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. The 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.

Keywords: API Specification, Interoperability, FAIR, Text Navigation

1. Introduction

Digital humanities projects and libraries publish digital collections of texts for many diverse reasons. Regardless of the original intent of publication, once published, the textual data itself could reach new audiences and support new research 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).

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 themselves are structured, or how the API itself is to be implemented. 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.

In this paper we will review related work in the realm of APIs for texts. We will discuss the origins of the DTS effort, the 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 the initial implementations of the API. We conclude with a discussion of the feedback received on the initial public draft and implementations and possibilities for future directions.

[1] https://www.w3id.org/dts

2. Related work

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 it was 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 (and possibly large) collection.

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 texts (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;
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.[3]

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[4], or the SHINE Open API specification (Wang et al. forthcoming; Ho et al. 2018; Wang et al. 2018)[5]. It is worth noting that, unlike other APIs including DTS, SHINE was developed with the support for licensed resources and their secured access as one of the key requirements.

There is one thing that all these APIs do, each with its own custom conventions, namely to enable the interoperability between digital text collections available on the Web. At the same time, by not agreeing on a common standard, they contribute to creating 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)[6]. 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

DTS was motivated by a desire to open to a wider range of texts and disciplines the possibilities that CTS made tangible for the subset of classical texts that could adhere to its 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 unavailable for the extensions needed to meet the core set of 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.

[2] http://cite-architecture.org/cts/

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

[4] https://github.com/digicademy/svsal/blob/master/docs/API.md

[5] https://rise.mpiwg-berlin.mpg.de/

[6] https://iiif.io/

These requirements were fleshed out in use cases and user stories. These can be found in the organization's GitHub repository, in the issues lists and wiki[7]. In development of the initial draft specification, two decisions in particular were difficult on which to come to consensus on.

The first was which standard to use for expression and documentation of the API. We wanted to use a standard that would (1) facilitate our own adherence to best practices and (2) facilitate adoption through widely available tooling to create and consume the API. The two main standards considered were the the Open API Specification[8] (OAS) and the Hydra Specification[9] (Hydra). 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.[10]

The second issue was whether or not to require TEI XML[11] as an output format for the API endpoint which 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). 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 a primary output format 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. That does not prevent repositories from making other formats available *in addition* to TEI. Link headers can be used in these cases to notify clients that other formats are available.

4. The specifications

4.1. General architecture

To fulfill the requirements, the general architecture of the DTS specifications revolves around 3 resource models (see Fig. 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.

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.

To take an example from the Alpheios Project's implementation of the DTS API[12], if the Greek edition of the *Histories* of Herodotus is available as urn:cts:greekLit:tlg0016.tlg001.perseus-grc2, 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.

An example from the Beta maṣāḥəft Project[13] shows how a different sort of collection can be served in the same way: in the case of a manuscript transcription having the id urn:dts:betmasMS:BLorient718, the collection resource attached to this identifier

[7] https://github.com/distributed-text-services/specifications

[8] https://swagger.io/specification/

[9] https://www.hydra-cg.com/spec/latest/core/

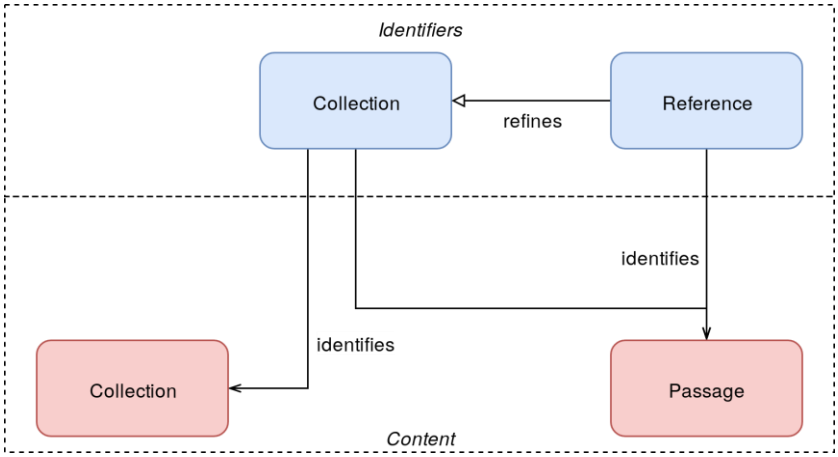
[10] On the simplicity of JSON-LD as one of the key success factors of linked data systems, see (H. A. Cayless 2019).

[11] https://tei-c.org/

[12] https://texts.alpheios.net/api/dts

[13] https://betamasaheft.eu

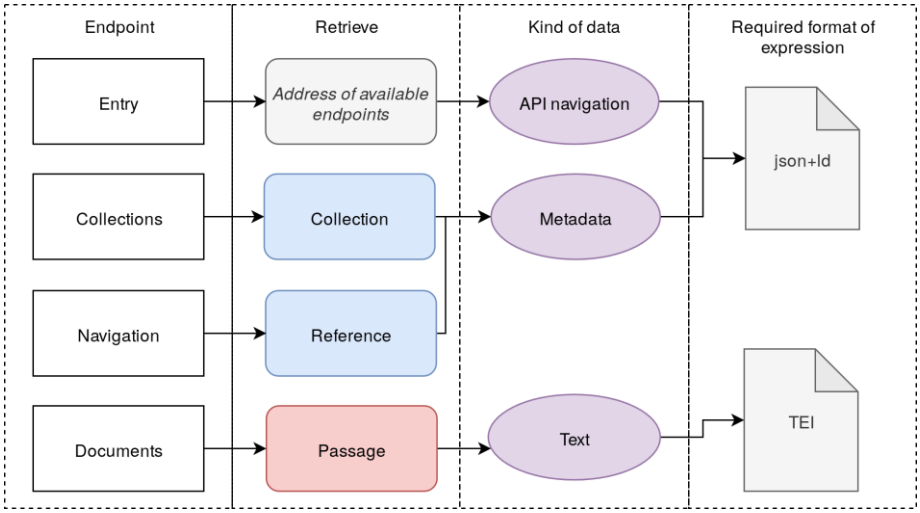
(https://betamasaheft.eu/api/dts/collections?id=urn:dts:betmasMS:BLorient718)[14] will contain metadata about the manuscript (shelfmark, cataloguer, repository) and the passage resource (https://betamasaheft.eu/api/dts/document?id=urn:dts:betmasMS:BLorient718) will contain its actual full text. If this text can be represented in segments, like folia, columns, etc., then references (e.g. https://betamasaheft.eu/api/dts/document?id=urn:dts:betmasMS:BLorient718) will provide metadata and segment identifiers for them.
And 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).



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

To link these 3 resource models and make them available, DTS provides specifications for 3 endpoints (or routes) that are respectively called Collections, Documents, Navigation (see Fig. 2). In order to be compliant, the DTS API specification does not require that all three endpoints are implemented. 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 However, the mandatory Entry endpoint is required: it declares and defines which other endpoints are available.[15]
Throughout, the API reuses standard HTTP methods and URIs[16] for its interaction with the client: the GET method is used for reading content, URL query parameters identify resources and filter properties, HTTP status codes are used as appropriate.
The API uses a namespaced 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.

[14] 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.
[15] This also means that projects which are in constant change and are 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 in them, so that it is possible to know that there is a passage Ex. 1.2, although no text is available for it (and the document API will tell).
[16] https://www.w3.org/Protocols/rfc2616/rfc2616.html



2. 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

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. The catalog can be flat (where each collection at the root is readable), tree-based (where each collection can only have 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.

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

The Hydra vocabulary allows us to differentiate between two types of collections, whose types are 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.

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. The following example 1 shows how such an extension can be used.

```
{
  "@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
}
```

3. Example Query 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.

4.2.1. Case Study 1 - Different recensions of a text

It is common that texts have different recensions even within one tradition. That is, the text was copied and altered and remains that text with such substantial changes to make it independent from other parallel recensions of the same abstract text in the manuscript transmission. 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 SAW[17] and CIDOC[18] ontologies are used in dts:extensions. In this example from https://betamasaheft.eu/api/dts/collections?id=urn:dts:betmas: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/LIT1401 Physio/title/t1" ]
   },
```

And, as in this example from https://betamasaheft.eu/api/dts/collections?id=urn:dts:betmas:LIT4915PhysA, each version will contain a statement to link it back to this general record, as well as a list of witnesses. The witnesses are expressed using dts:dublincore following the suggestion made in (H. Cayless and Romanello forthcoming).

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

4.3. Navigation Endpoint

The Navigation endpoint answers the need to provide an index of available passages for a text when this capacity is possible. It enables a client to provide browse functionality and supports complex hierarchies in which each browsable passage may itself contain distinct passages of varying numbers. 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 browse purposes are better off grouped in predefined chunks).
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.
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

[17] http://www.ancientwisdoms.ac.uk/
[18] http://www.cidoc-crm.org/

data sources and software stack[19]. 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[20].


```
{
  "@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. Example query 2: On the TNAH endpoint, request for the reference between 1 and 10 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

4.3.1. Case Study 2 - Excerpts only

The text entitled *Malkəʾa Gabra Manfas Qəddus* In the Beta maṣāḥəft DTS collection has been identified in some manuscripts from Dayr as-Suryān in Egypt, but in the process of cataloguing the manuscripts only 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 only added a label for the first stanza, not for both those from which some text was copied. The Navigation endpoint reflects 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

Horace's *Ars Poetica*[21] is a single poem, has a simple one-level navigation hierarchy, with one reference per line:

[19] For example, if the implementers were using self closing xml elements in xml source files for their application, range and reference in general can quite difficult to implement if the stack does not include XPath 2.

[20] 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.

[21] In the Perseids DTS API at https://dts.perseids.org/navigation?id=urn:cts:latinLit:phi0893.phi006.perseus-lat2.

```
{
  …
  "@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" }
  ]
}
```


Whereas Horace's *Epistulae*[22], a work composed of individual books and poems, has a three-level hierarchy, given that poems are in turn divided up into poetic lines. 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

[22] In the Perseids DTS API at https://dts.perseids.org/navigation?id=urn:cts:latinLit:phi0893.phi005.perseus-lat2.

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 (cf. supra), however, it might be important for implementers to also offer plain text, html or alto-xml or other representations of requested passages.
As a primary motivator for the 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. For this reason, the Documents endpoint is the only endpoint that uses headers to convey navigation information: links to collections and navigation endpoints, as well as next, previous, parent passages link.
In order to support retrieval of fragments of text, the DTS API introduces an XML tag under the DTS namespace, 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 result passage is a fabricated document.

```
<?xml version="1.0" encoding="UTF-8"?>
<TEI  xmlns= "http://www.tei-c.org/ns/1.0"
  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. Example query 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

5. Existing implementations and APIs

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

- Small corpora
  - * Ecole Nationale des Chartes covers two periods for its corpora: contemporaneous and medieval. Contemporaneous are lightly marked up, while medieval corpora finely annotated[23].
  - * Alpheios.net provides a small collection of Latin and Greek texts which have been aligned with linguistic annotation for learning ancient languages[24].
- 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[25].

[23] https://dev.chartes.psl.eu/api/nautilus/dts
[24] https://texts.alpheios.net/api/dts
[25] https://dts.perseids.org/

* Beta maṣāḥəft collects written artefacts from the highlands of Ethiopia and Eritrea mainly in Gəʿəz (Classical Ethiopic)[26]. 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 the Latin epigraphic databases[27].

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

- the TEI Publisher client, that 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, Munson, and Almas 2019).

Both MyCapytain through Nautilus[28] and the TEI Publisher offer a server API, with the same restrictions as the client for the last. Another implementation for small corpora is offered by the DTS-Demo-Server in Python with an SQL database[29]: it supports one level only in navigation (Dartois, Vieillon, and Clérice 2019).

6. Conclusions

Implementing a DTS-compliant API contributes to the efforts of publishers of text collections to adhere to both FAIR[30] (Boeckhout, Zielhuis, and Bredenoord 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 specification of relations among the resources
- has no canonical implications - relations between texts depend on the collection
- does not impose any requirements upon how the data is stored
- is extensible, open, free, and universally implementable.

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[31] and the Google Groups discussion list, which as of the writing of this paper has 45 members, 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 able to make a commitment to active engagement. Additional roles of Ambassador and Contributor have been defined to enable diverse types of contributions.[32]

The first draft specification was released in the fall of 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.

One need which falls a bit outside of the scope of the DTS effort itself is that for a centralized resolution service that could locate a text or texts from within a network of distributed DTS implementations (H. Cayless and Romanello forthcoming). Making this a reality will require community support for shared infrastructure and agreement on the vocabularies and best practices for metadata.

Notwithstanding the work that still 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 speed of progress towards a fully open, interoperable ecosystem of texts as linked, machine-actionable data.

[26] https://betamasaheft.eu/api/dts
[27] https://edh-www.adw.uni-heidelberg.de/api/dts/
[28] https://github.com/Capitains/Nautilus
[29] https://github.com/Chartes-TNAH/dts-demo-server
[30] https://www.force11.org/group/fairgroup/fairprinciples
[31] https://github.com/distributed-text-services/meeting-notes
[32] https://distributed-text-services.github.io/specifications/Organization.html

7. 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.” 2018. https://w3id.org/dts.

Almas, Bridget, and Caroline Schroeder. 2016. “Applying the Canonical Text Services Model to the Coptic SCRIPTORIUM.” *Data Science Journal* 15 (0): 13. https://doi.org/10.5334/dsj-2016-013.

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

Blackwell, Christopher W., and Neel Smith. 2019. “The CITE Architecture: A Conceptual and Practical Overview.” In *Digital Classical PhilologyAncient Greek and Latin in the Digital Revolution*. Vol. 10. Berlin, Boston: De Gruyter Saur. https://doi.org/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 (7): 931. https://doi.org/10.1038/s41431-018-0160-0.

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

Cayless, Hugh, and Matteo Romanello. forthcoming. “Towards Resolution Services for Text URIs.” In .

Clérice, Thibault, Bridget Almas, Hugh Cayless, Vincent Jolivet, Emmanuelle Morlock, Jonathan Robie, James Tauber, Jeffrey Witt, and Pietro Liuzzo. 2018. “From File Interoperability to Service Interoperability : The Distributed Text Services.” In *TEI 2018*. Tokyo, Japan. https://hal.archives-ouvertes.fr/hal-02196659.

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

Dartois, Hélène, Lucie Vieillon, and Thibault Clérice. 2019. *Chartes-TNAH/Dts-Demo-Server: V1.0.0*. Zenodo. https://doi.org/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*. Institute of Electrical and Electronics Engineers. https://doi.org/10.23919/PNC.2018.8579459.

Smith, Neel. 2009. “Citation in Classical Studies.” *Digital Humanities Quarterly* 3 (1). http://www.digitalhumanities.org/dhq/vol/003/1/000028.html#.

Turska, Magdalena. 2019. “TEI Publisher 5.0.0.” *TEI Publisher* (blog). August 2, 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). https://doi.org/10.1045/december2004-vandesompel.

Villa, Massimo. 2018. “Encoding Strategies and the Ethiopic Literary Heritage: The *Physiologus* as a Case Study.” In *COMSt Bulletin*, edited by Alessandro Bausi, Paola Buzi, Pietro Maria Liuzzo, and Eugenia Sokolinski, 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.” *DADH. 9th International Conference of Digital Archives and Digital Humanities*, December, 21–37.

Wang, Sean, Belouin Pascal, Hou Ieong Ho, and Chen Shih-Pei. forthcoming. “RISE and SHINE: A Modular and Decentralized Approach for Interoperability between Textual Collections and Digital Research Tools.” Text. forthcoming. https://dev.clariah.nl/files/dh2019/boa/0607.html?origin_team=T5DKR7J74.

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 (March). https://doi.org/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. http://www.uni-koeln.de/.