CTS, EpiDoc, and the CapiTainS Testing Environment

Lenny Muellner

Director for Publications and IT, Center for Hellenic Studies

1. Overview of CTS

- CTS (Canonical Text Services) 3.0: system for referring to, resolving, and retrieving references to any text or any point in a text that is part of a CTS-compliant corpus of texts.
- What makes a CTS text CTS-compliant?
- Notion of logical (vs. presentational) structure: text reference as a page number or range in a printed book, vs. text reference as a point or range in a logical structure
- CTS requires a logical structure, or a page reference that can be converted into a logical structure: Herodotus Book 3, Chapter 2, Section 4 aka Herodotus 3.2.4

CTS URN and Internal Map of Text Structure

- Two key features to make text reference and retrieval possible
- Standardized URN (Uniform Resource Name, an alternative to HTTP) that points to the text file that contains the text being referenced
- Map of text's structure inside the text itself, so that the system can navigate to the point or the range being referenced (Xpath)
- What constitutes a valid CTS URN?
- A look at an internal structural map

Valid CTS URN

An example of a valid CTS URN referring to the first letter eta of the first sentence of Herodotus' Histories:

urn:cts:greekLit:tlg0016:tlg001:1.1.1@η[1]

1. URN namespace – here, cts
2. CTS namespace – here, greekLit
3. Work identifier – here, tlg0016 (Herodotus textgroup in the TLG Canon), tlg001 is the Histories (in the case of Herodotus, there are no other works in the textgroup)
4. Optional passage reference, with a dot (.) separating the secondary level elements, here book, chapter, and section
5. Optional subreference, after the @ to Unicode Greek codepoint of letter or a word, in this case a word, and in square brackets, which occurrence of that element, in this case, the first

CTS-Compliant Internal Map Structure

Embedded in the XML header of the text, an Xpath expresion that traverses the text as a logical structure::

```
<refsDecl n="CTS">

        <cRefPattern n="section" matchPattern="(\w+).(\w+).(\w+)”
replacementPattern="#xpath(/tei:TEI/tei:text/tei:body/tei:div/tei:div[@n='$1']/tei:div[@n='$2']/tei:div[@n='$3'])">
           <p>This pointer pattern extracts Book and Chapter and Section</p>
        </cRefPattern>

        <cRefPattern n="chapter" matchPattern="(\w+).(\w+)"
          replacementPattern="#xpath(/tei:TEI/tei:text/tei:body/tei:div/tei:div[@n='$1']/tei:div[@n='$2'])">
           <p>This pointer pattern extracts Book and Chapter</p>
        </cRefPattern>

        <cRefPattern n="book" matchPattern="(\w+)"
          replacementPattern="#xpath(/tei:TEI/tei:text/tei:body/tei:div/tei:div[@n='$1'])">
           <p>This pointer pattern extracts Book</p>
        </cRefPattern>

</refsDecl>
```

This is complex, and not necessary to understand at this point, but it maps out the structure of the text for references to a book, a book and a chapter, or a book, a chapter, and.a section

2. EpiDoc (part 1: elements)

- EpiDoc is a subset of TEI (Text Encoding Initiative) XML, a markup standard for documents in the Humanities
- XML is an international standard for marking up documents that features elements with attributes and entities (we'll postpone entities for the moment)
- An element is a standardized name for a structural feature (like a section <div>…</div> or a paragraph <para>…</para>) or a semantic feature (like the title of a work <title>Nightwood</title>) of a text.
- Elements have one or more attributes that further refine them: a section of a document, a <div>, can be a book or a chapter, so it can have an attribute 'subtype' with a value of either 'book' or 'chapter':

  <div subtype="chapter">…</div>

EpiDoc (part 2): Syntax

- An XML document references a set of rules that define its elements as well as the way that they go together (there are several acceptable formats for this specification)
- EpiDoc consists of a subset of TEI XML elements and their attributes, but it also has a syntax that is appropriate to them. You can’t have a <div subtype=“book”>…</div> inside a <div subtype=“chapter”>. That isn’t logical, so the rules that are attached to the document specify how the elements can be assembled and nested.
- These rules (which can be formulated in XML also) allow for the validity of the document – in other words, its use of the allowable element names and attributes and its logical structure – to be tested by software.

CapiTainS: How to test a document for CTS- and EpiDoc-compliance

- The CapiTainS environment is a suite of tests that checks the validity of a file or a whole Github repository for CTS- and EpiDoc compliance.
- It runs a test suite called Hooktest on a document or a set of documents (even a whole repository), and it sends back the results to its user.
- CTS-compliance means having a proper internal structure that corresponds to the code for its mapping as well as being a correctly formed EpiDoc text.
- Hooktest generates errors that require some interpretation. For instance, it can report errors in “Passage level parsing”, which means that the Xpath expression that maps the document’s structure is incorrectly formed.
- Working with these tools means identifying errors, fixing them, and iteratively running Hooktest on Travis until no errors are generated.