Writing Fragment Identifiers

Revision History
2021-06-27
Prepublication draft, version 0.02

Table of Contents

Abstract 2
Introduction 2
Conformance 3
    Media Formats 4
    Processors 4
Motivation and Goals 5
Challenges 7
Concepts and Definitions 8
    Writing, Works, Scripta, and Shared Limits 9
    The Relationship between Works and Scripta 11
    Scripta: Readers, Regions, and Text Structures 11
    The Trunk Text: Divisions, Units, Sequences, and Trees 13
    Labels; Reference Units, Sequences, and Systems 15
General Stipulations 18
Syntax 19
    Base URI 19
    Parameters 21
    References 22
Strategies for Writing a WF URI 24
Examples 28
Scrap 28

Warning

This prepublication draft document is subject to major revisions. Requests for changes should be directed to the editor, either by email (kalvesmaki@gmail.com) or by GitHub ticket.

Things to do:

- Enlist specialists to critique this document.
- Discuss questions raised throughout.
- Develop a suite of about 100 examples drawn from a range of types of reference systems.
- Write XSpec test suite.

Warning

Look throughout the document for warnings, which register significant doubts the editor wants to leave for discussion, deliberation.

Warning

For about eight years I've been thinking about reference systems and pointers. That thinking has shaped the design of Text Alignment Network [http://textalign.net] (TAN) XML, but I have adopt-

ed a somewhat different perspective in writing these specifications, because my primary design concern has been not a data format but an unambiguous, clear URI fragid. That is, the problem I've set out to solve is rather different, so I've taken a different approach. In fact, having finished the first draft, I have realized that TAN is not completely WF-conformant. In fact, it is somewhat easier to define a customization of TEI for WF conformance.

Abstract

This document defines the syntax for a fragment identifier ("fragid") to any URI that designates a piece of writing. Such *Writing Fragids* (WFs) enhance the URI, by allowing anyone to cite a particular part of a work, and providing a method for organizing, retrieving, and disseminating digital content in a variety of media and formats.

These specifications for WFs define a semantic fragid structure, and not a method for negotiating a single media type. They explain how to construct WF URIs, and establish conformance requirements, both for media formats and for applications that extract content.

To construct a fragid syntax that restricts a reference unambiguously to a specific part of a piece of writing has numerous challenges. Writing fragids were designed to model habits of citation, which have developed in complex ways over centuries. WF specifications isolate a subset of citation practices and correlate it with a syntax that enables the creation of a fragid that is just as persistent and unique as the URI it modifies.

Introduction

Numerous Universal Resource Identifiers (URIs) designate pieces of writing. Some point to a writing in the abstract without reference to a particular version or edition (e.g., http://dbpedia.org/resource/Iliad for the *Iliad*), whereas others point to a particular written artefact, without reference to what kind of writing it is (e.g., urn:uuid:61c31a6d-c45a-4093-b25f-5a9c94566172 for the receipt in my pocket). In either case, someone may wish to use the URI not merely to name the writing, but to designate a specific range, place, or portion of the writing. One practical way to do so would be to add a fragment identifier ("fragids") to the URI.

This document defines the rules for *Writing Fragids* (WFs), designed to be applied to any URI that designates either an abstract written work or an object that has writing on it. Although WFs may be applied to any type of writing, it has been designed primarily for those that are objects of citation, and survive in non-digital media (e.g., books, articles, newspapers, manuscripts, inscriptions, papyri, ostraca).

Warning

Currently, the WF specifications are written claiming to support all types of scripta and works, later making exceptions (opt-out). While writing the examples, I thought about an alternative approach, namely, to predefine a few of the most common reference patterns, and say that no scriptum or work could be a candidate for WFs unless it conformed (opt-in). I can see arguments pro and con for either approach.

1. Opt-out: a general set of rules applicable all the time for every type of text, regardless, with some exceptions excluded. If there are inconcinnities, then the hell with it. Anyone who encodes a WF URI / WF-compliant text that relies upon an idiosyncratic understanding of a scriptum or work, well, that's on them, and WFs will be ignored because they are not approaching the material the same way everyone else is. This approach, adopted for in this version of the specs, would keep the number of rules to a minimum.

2. Opt-in: rules require that anyone who encodes a WF URI / WF-compliant text declare such-and-such a predefined WF reference system type, and don't worry about unsupported texts. As the WF specifications develop, with new versions of the WF specs, important reference

systems can be added to increase the types of supported writings. This approach would require a lot more rules declared at the outset.

I do not have a clear sense on which way the needle should point.

URIs with WFs enable two kinds of activity that are otherwise difficult or impossible:

1. **Shared URI-based citations to specific parts of a work**. Rather than pointing to an entire work, one can specify a particular part. This in turn allows one to build computer-actionable statements that are more precise. One can build such WF-URIs without depending upon a particular digital surrogate. In fact, one can coin and use WF URIs for writings that have not yet been digitized.

2. **A URI-based protocol for requesting and delivering digital resources**. A WF-URI is media independent, and may correspond to text, image, audio, video, or other types of resources. The WF specifications permit designers of data formats to allow users to expose their data to WF URIs, and developers of applications to retrieve and deliver requests that include WFs.

These guidelines adhere to the Best Practices for Fragment Identifiers and Media Type Definitions [https://www.w3.org/TR/fragid-best-practices/], specifically the section addressing fragid structures [https://www.w3.org/TR/fragid-best-practices/#structures]. A semantic fragid structure such as WF declares a set of meaningful syntactic rules that can be followed by designers of individual media types, perhaps to be registered with the Internet Assigned Numbers Authority (IANA) [https://www.iana.org/].

Fragids in general allow processors to extract specific content from digital resources. Writing Fragids are also intended for those purposes, but on an abstract level, without specifying exactly which digital resources, if any, will match the WF. Indeed, some WFs will be added to URIs that have no specification on how or where to retrieve a matching document. Further, many WFs will be attached to URIs that identify nondigital textual entities that may not have any corresponding digital resource for years to come, if ever. Such resource-agnostic URIs are valuable, because they allow one to make assertions about nondigital texts without having to rely upon any particular digital surrogate.

Designers of WF-conformant media types must write their own guidelines defining exactly how the individual components of a WF URI must be interpreted and resolved against instances of the format. Developers creating parsers or applications handling a specific WF-conformant data format must take into consideration both these guidelines and the ones stipulated by any WF-conformant data format.

These specifications engage with technical material from two different areas: writing technology and computer science. Some readers knowledgeable in one but not the other may find reading sequentially through this document to be frustrating and confusing. The following order is recommended for new readers: the section called “Motivation and Goals”, the section called “Challenges”, the section called “Examples”, then other sections as interest leads.

Conformance

Important

This section is normative.

The key words *must*, *must not*, *required*, *should*, *should not*, *recommended*, *may*, and *optional* in this specification are to be interpreted as described in RFC2119 [https://www.w3.org/TR/fragid-best-practices/#bib-RFC2119].

Warning

This section is a bit of a hodge-podge, a mixture of ideals and harsh realities faced when trying to write my conformance suite. It is also probably excessive. As I've been developing both a media format and a processor, I've noted what I've wanted to achieve.

Media Formats

[Definition: *WF-conformant media formats* are media formats that have been designed to allow files in the format to be parsed against WF URIs for content.] In defining conformance of a media type to the Writing Fragid specifications, designers must provide a narrative describing how to match the component parts of a WF URI against a file, and how the file should be parsed to extract matching content. Although not required, it is recommended that designers include one or more algorithms, along with a test suite, to corroborate the narrative programmatically.

Every WF-conformant media format *must* stipulate in its WF guidelines which version of WF is supported.

WF was designed to support textual scholarship, which regards provenance as being highly important. In many cases, when content is extracted from a WF-conformant file, the recipient of the data will expect metadata that stipulates responsibility. Therefore, every WF-conformant media format *must* define a mechanism that supports provenance information. Whether such information must always be included, or how it is validated, is left to designers of the media format.

The WF syntax is not universally comprehensive. Some of the content in a WF-conformant file may not be accessible to a WF URI. It is not required that all the content in a matching WF-conformant file be accessible by a WF URI. Guidelines for WF-conformant media formats must explain how to distinguish WF-qualifying content from non-WF-qualifying content.

Some generic media formats (e.g., TEI XML, HTML) may be conducive to WF-conformance, but may also permit multiple ways of being customized. Anyone customizing a generic media format to be WF-conformant *should* provide a unique name for the customization, and *should* develop mechanisms that avoid conflict with any other existing WF-conformant definitions for the same generic format.

Some media media formats might be designed not to expose content for extraction via WF URIs, but to be a generative source of WF URIs (e.g., a file format designed for RDF triples that include WF URIs, or for data structures that can be converted to WF URIs). No conformance requirements are stipulated of such WF-engaged formats beyond the WF syntactic requirements defined below.

Other conformance requirements for media formats appear throughout these specifications.

Processors

[Definition: A *WF processor* is defined as an algorithm that takes as input one or more WF URIs and one or more files or file fragments from WF-conformant media formats, and returns as output the media content that matches the WF URIs, perhaps along with associated metadata.] A WF processor may be written in a variety of programming languages.

WF processors may feature, depend upon, or actually be, an application programming interface (API) or some other type of interface. The WF specifications put no strictures on such interfaces, aside from its processing requirements.

If a WF is added to a URI that begins with the regular expression `https?://`, the URI is to be interpreted not as a URL per se (a place where a particular digital resource is to be found) but as the identifying name of a piece of literature. A WF processor *may* treat such a URI as a location and attempt to retrieve data from it, but these specifications place no strictures on the mediating web protocol or interface, or what type of resource might be returned by a referenced server, or how to interpret the content that is returned.

WF URIs were created to support provenanced claims about texts. It is *recommended* that WF processors provide in any output metadata that identifies the processor and its version and any provenance metadata associated with extracted content.

A WF processor *must* expect input that conforms to the content format defined by the target WF-conformant media. That processor *may* apply further adjustments before it renders output. The format

and serialization of finalized output WF data is implementation-dependent, but the representation of the content of WF media format content *should* be lossless. Those who design WF processors will likely need to constrain the output to a particular format (XML, JSON, tiff, mp4, svg, etc.).

Warning

Does this paragraph get too invasive? Those who write processors will do whatever they want, in the end.

A WF processor *must* define the types of WF conformant-media it processes. A WF processor need not support all media types, but it *must* support all syntactically valid WF URIs.

If a WF URI constructs reference nodes in a specific sequence, the corresponding output *must* respect the same sequence.

Error reporting is implementation-dependent, unless otherwise specified.

A non-normative companion XSLT test suite accompanies these specifications. It can be used to parse and validate WF URIs, or to build analogous processes in other programming languages.

Warning

Currently under construction at wf-functions.xsl [./wf-functions.xsl].

Other conformance requirements for WF processors appear throughout these guidelines. Designers of WF processors *must* document how implementation-dependent decisions are handled.

Motivation and Goals

Important

This section is descriptive.

The need for Writing Fragids has grown with the influence of the Semantic Web, an ecosystem of sharing data interoperably, based on the model defined by the Resource Description Framework [https://www.w3.org/RDF/] (RDF). RDF defines a relatively simple graph model for its basic datum, called a triple, a graph component that consists of two nodes and one edge, named the subject (node), predicate (edge), and object (node). As the names suggest, a RDF triple models an everyday claim.

In the case of written texts, one might wish to say, "Plato's *Republic* quotes from Homer's *Iliad*." Converting such an assertion into RDF is at present relatively trivial (adopting Turtle syntax, the most readable RDF serialization method):

```
@prefix db: <http://dbpedia.org/resource/> .
@prefix cito: <http://purl.org/spar/cito/> .

db:Republic_(Plato) cito:cites db:Iliad
```

Or, to take an example from modern literature, the statement "Walter Burkert, in *Lore and Science in Ancient Pythagoreanism*, quotes from Plato's *Laws*" can be reduced to a RDF triple as follows:

```
@prefix db: <http://dbpedia.org/resource/> .
@prefix cito: <http://purl.org/spar/cito/> .
@prefix wc: <http://www.worldcat.org/oclc/> .

wc:860129739 cito:includesQuotationFrom db:Laws_(dialogue)
```

This could apply as well to journal articles. The following claims that Sadrine Zufferey and Bruno Cartoni, "A Multifactorial Analysis of Explicitation in Translation," *Target: International Journal of Translation Studies* 26.3 (2014): 361-384 cites Kinga Klaudy and Krisztina Károly, "Implicitation in

Translation: Empirical Evidence for Operational Asymmetry in Translation," *Across Languages and Cultures* 6.1 (2005): 13-28:

```
@prefix doi: <https://doi.org/> .
@prefix cito: <http://purl.org/spar/cito/> .

doi:10.1075/target.26.3.02zuf cito:cites doi:10.1556/Acr.6.2005.1.2
```

RDF triples that refer to complete works are rather straightforward. But such assertions, even if true, are too general to be useful. The citations and quotations by Plato, Burkert, and Zufferey + Cartoni are made *at* specific pages, *of* specific passages. Those who desire greater precision would find more helpful sets of RDF triples that specify exactly where one item quotes the other, i.e., at page A line B book X quotes from section C subsection D of work Y.

In making a triple more precise, the subject and object would remain the same, but narrowed in scope to specific parts. The endeavor is analogous to current conventions that allow one to point to a specific section of an individual image, video, audio file, web page, XML file, and so forth. Many of these file-specific conventions depend upon URI fragment identifiers that reference content within the resource. The present specification for Writing Fragids proposes an analogous construction, for URIs that point, not to digital resources, but to documents, publications, literature, and other types of writing that lend themselves to referencing in general. Just as a URI can point to a non-digital entity, so to can a WF.

A URI with a Writing Fragid enables RDF triples that point with greater precision to written literature in any language, from any period of time. WF syntax does not depend upon any particular digital service or digital file, so WF URIs can be made and used, regardless of how many corresponding files exist, if any. The WF syntax allows a person or algorithm to make a claim that points unambiguously to a particular location in a work or item. That assertion, whose meaning is independent of any digital resource, can then be used to extract relevant sections from any matching WF-conformant files, and perhaps to make other inferences not envisioned by the creator of the original WF URI or RDF triple.

Although the preceding use cases have focused on quotation, a WF URI may be used in RDF triples for a variety of reasons, for example, to make assertions about textual variation, dates, authorship, or even grammatical properties. Consider the statement, "In Shakespeare's *Henry VI*, part 2, 1.4.32, 'Henry' is the grammatical object."

Note

The statement is reductive and not obviously true; the context is, "The duke yet lives that Henry shall depose," a cunning amphiboly with two very different intepretations based upon whether one interprets "Henry" as the grammatical subject or object of "depose".

A WF-based RDF triple encoding this assertion might be built as follows (with a placeholder for the Writing Fragid component):

```
@prefix db: <http://dbpedia.org/resource/> .
@prefix la: <http://example.org/linguistic-annotation> .
@prefix olia: <http://purl.org/olia/olia.owl#> .

db:Henry_VI,_Part_2#[WRITING_FRAGID] la:hasFeature olia:DirectObject
```

These guidelines stipulate syntactic rules for constructing a WF such that the modified URI remains persistent and unique, and therefore unambiguous.

Persistence and uniqueness are foundational requirements for any URI. Something may be assigned many URIs, perhaps by even the same person or organization, but no person should wittingly assign one URI to two or more different resources (or classes of resources), or to create a URI that could be interpreted in mutually exclusive ways. The same should hold for URI fragids.

The WF syntax has been designed to avoid ambiguity, at the expense of some brevity and legibility. Anyone who learns the syntax can write and read a WF URI, but its full meaning may not be apparent *prima facie*.

Writing Fragids have been designed to model citation practices, which have challenging features (discussed below). Writers frequently refer to other writings in a terse syntax that permits multiple interpretations. They take for granted that the reader comes with a shared context to understand and resolve a citation. Such tacit knowledge has been reliable in writing for human understanding. But URIs, which are required for computer "understanding", must be based upon explicit and unambiguous conventions. Because WF URIs are to be persistent and unique, traditional citation practices that are at present not easily convertible are excluded from this version of the WF specifications.

Challenges

Important

This section is descriptive.

A Writing Fragid is sequence of characters (letters, numbers, punctuation) that models the core features of traditional references. Reference systems are essential and ubiquitous, but poorly studied and understood. An overarching theory or framework of citation practices is elusive. As of this writing, there does not even exist a relevant Wikipedia entry on the subject. Many reference systems feature a number of irregularities that challenges any effort to create a WF URI that is persistent, unique, and interoperable.

Many texts have more than one reference system. For example, if you are asked to look up Aristotle, *Categories* 4, do you go to chapter four, or to Bekker page number 4? It is impossible to tell, since that particular text has two major reference systems, and the reference is valid in both systems. If you encounter a reference to the Bible verse Joshua 9:2a, it is clear that one means to point to chapter nine, verse two, but does the "a" point to the first part of verse two (and how many parts are there?), or does it mean verse 9a, i.e., an extra verse between Joshua 9:2 and 9:3? All three options are possible, because Joshua 9:2 can be divided however one likes, and in the best critical editions of the Septuagint version of the Old Testament, extra verses appear between Joshua 9:2 and 9:3 and are given by their editors subletters (Joshua 9:2a, 9:2b, etc.).

Most human readers familiar with the conventions for a given work can quickly and easily interpret terse, vague, and ambiguous syntax, but computers cannot. Any algorithm that currently negotiates such differences has been extensively programmed, usually on the basis of a small set of controlled vocabulary and a limited set of assumptions. No algorithm has been written to disambiguate any reference to any body of literature.

One may argue that one should adopt the reference system most people agree upon for a given work. Although this proposal, based on the notion of a so-called canonical reference system, is well motivated, it is inadequate, even for very common examples. As noted above with Aristotle's *Categories*, many written works have two or more independent reference systems (e.g., in a book, chapter/section numbers versus page/line numbers, e.g., Plato's corpus). Texts with an unambiguous canonical reference systems are rare. Even within the very text collection that gave us the notion of "canon", the Bible, there are standard reference systems that conflict with each other. For example, the reference Psalm 30.1 points to three different passages, depending upon whether the reference is interpreted as being reliant on English, Hebrew, or Greek Septuagint versification:

Table 1. Psalm 30.1

English (KJV)	Hebrew (JPS)	Septuagint (NETS)
I will extol thee, O LORD; for thou hast lifted me up, and hast not made my foes to rejoice over me.	A psalm of David. A song for the dedication of the House.	Regarding completion. A Psalm. Pertaining to Dauid.

In this case, Psalm 30.1 English = Psalm 30.2 Hebrew = Psalm 29.2 Septuagint. Roughly 10% of the verses in the Tanakh/Old Testament are subject to such reference variants in the three major versification schemes used today.

There are many, many other works with comparable discordant, ambiguous, or vague reference systems. For example, many classical works are assigned specific references corresponding to the page, column, and line numbers of the first critical edition (e.g., Bekker numbers for Aristotle, Stephanus numbers for Plato). But without exception, every 20th- and 21st-century edition and translation of texts that adopts these edition-specific systems must adapt and adjust those systems. A particular word might fall before, after, or across, a particular line number.

Another type of challenge pertains to the labels used in reference systems. Many reference systems rely on numerals, or ordered letters that enumerate textual units. But many other labels are not enumerated, and reference conventions differ widely. To draw again from Biblical literature, reference terms for books are not standardized. The book commonly called 1 Samuel has the name 1 Kingdoms in the Septuagint, and 1 Kings is 3 Kingdoms. When one refers to the book Ezra, any number of books might be meant, across a range of modern national or religious traditions. Some references are given in full, others in abbreviated form. The latter increases the chance of confusion. For example, "Jo" could mean the Biblical books John, Job, Joel, or Jonah. When expanding to other languages, the number of potential problems grows exponentially.

Many generic textual units normally do not admit number labels but are referred to by their type. For example, one may reference a title through "title", "ti", "t" or the like, presenting choices that could lead to confusion. Does "ep" refer to an epilogue or an epistle; does "no" mean note or number? Some text components are difficult to name and cite (e.g., parts of a liturgical text). As a result, labels for, and references to, special types of textual divisions are created ad hoc, according to custom, but not any controlled vocabulary.

Even enumerated sequences are not straightforward. As noted above, with Joshua 9:2a, some references depend upon compound numbers. Different types of numeration systems exist, e.g., Roman numerals and letter labels. Although these can invariably be converted into Arabic numerals, individual cases may not be clear without an explicit context. For example a reference to "cc" could mean 200 if pointing to a text that uses Roman numerals, and either 29 or a different number if using letter labels (some letter label systems assume incomplete alphabets, or have different rules for enumerating beyond the twenty-six items of the English form of the Latin alphabet).

In short, reference systems present numerous problems. A variety of conventions have developed independently over centuries, across languages and disciplines. We divide, label, and refer to each others writings in a bewildering variety of ways. We often have a difficult time negotiating such differences; computers do not fare better.

Concepts and Definitions

Important

This section is normative.

The definitions below are normative only within the scope of WF URIs. One should not infer from the terms or their definitions that they stipulate the best way to think about written texts.

Warning

This section is overly long, and I admit steps should be dropped. I've erred on the side of excess for this first draft, because previous attempts to deal with this task have been, in my opinion, poorly theorized. I think the downfall of FRBR and OHCO is that their definitions assumed of the reader too much. They were not been approached incrementally, from the simplest ideas. Hence my excessive approach.

Much of the discussion is technical and difficult to follow. Because this is ultimately a technical document, and needs to be rigorous, I would personally prefer to keep the rigor and provide a second "Gentle Guide to Making a WF URI" in a casual tone suited to teaching.

As I re-read my draft, I blame myself for overthinking the matter. But I haven't deleted a lot, so as to leave at least a record of some early thoughts, and to catalyze a wide-ranging discussion among those who can do better.

Writing, Works, Scripta, and Shared Limits

Writing

[Definition: *Writing* is defined as anything a human creates through visible marks, i.e., characters and symbols, to communicate ideas and meaning to later readers.]

[Definition: An *item of writing* is an instance of writing.] Every item of writing must be referred to *qua* conceptual/immaterial entity, or *qua* instantiated/material entity. For example, one can talk about the item of writing called the *Iliad* without reference to any particular edition, or one can talk about a specific printed edition of the Homeric poems. Items of writing discussed in the first mode are termed *works*; in the second, *scripta* (singular: *scriptum*). The distinction is to be made even when it is not so evident (see below).

Works

[Definition: A *work* is a conceptual, nonmaterial item of writing. Every work is a notional entity with shared limits, whose content admits division (perhaps into other works), and whose content cannot be equated or conflated with any individual material entity.] Homer's *Iliad* and Shakespeare's *Henry VI* are examples of conceptual works: they are items of literature that can be meaningfully discussed independent of any of the many individual books or manuscripts that carry editions, translations, or other versions of these works.

Works frequently encompass other works (e.g., the Catalog of Ships or Gloucester's Soliloquy), or they may be constituent parts of larger works or collections, which are also treated here as works (e.g., the Homeric Cycle, Shakespeare's minor tetralogy). Work-to-work relationships may exhibit multiple levels of nesting, repetition, or interconnection. For example, the work *The New Testament* encompasses the work *The Gospel of Matthew*, which includes the work *The Sermon on the Mount*, which includes the work *The Lord's Prayer*, a work that is itself a constituent part of many other works, sometimes repeatedly so (e.g., a Christian liturgical service).

Any single work may have derivative versions (e.g., translations, paraphrases, parodies). Such work-versions are to be treated as works in their own right. [Definition: To every work-version may be attached one or more *version paths*, defined as the chronologically ordered sequence of work-versions upon which a particular work-version depends.] A version path may be unclear, or disputed, e.g., texts preserved in medieval manuscripts hundreds of years removed from any putative original, with perhaps questions of contamination. Version paths of more recent works might be quite clear. For example, a modern critical edition (a work-version in its own right) collectively extends the length of the version paths of its manuscripts by one, and a translation of that critical edition's work-version into a modern language has a version path yet one step longer.

If a work-version is the subject of a version path, the name of that work-version may be applied not merely to the original work-version, but to all work-versions along each of its resultant version paths. The name "Lao Tzu's *Art of War*" applies to all its versions. Hence, the name of a work-version should be treated as labeling not a single item but a class of items. That class may be restricted to subclasses. Any such restriction may be made on the basis of a version path or not. "Lionel Giles' 1910 translation of Lao Tzu's *Art of War*" uses the a version path to restrict the original class. "Twentieth-century versions of Lao Tzu's *Art of War* published in the United States" uses criteria not reflecting any particular version path.

The name of a work-version is inheritable by its versions (a transitive, asymmetric relation). West's critical edition of the Greek text of the *Iliad* is a particular version, but it can be legitimatly given other more general names: (1) West's edition of the *Iliad* in Greek; (2) the Greek version of the *Iliad*; or

simply (3) the *Iliad*. The last of these names can also be applied to an English translation of West's Greek edition, which could also be properly termed (4) West's version of the *Iliad* (the language is not specified); (5) an English translation of the *Iliad*, or (6) an English translation of West's edition of the *Iliad* in Greek. Other names, following certain permutations, could be used. Although the examples above illustrate the point with human-readable names, it applies as well to machine-readable ones, such URIs.

Warning

The previous two paragraphs exhibit some overthinking on my part, but they also reflect a real problem I discuss elsewhere, about how to point to a work-version in particular, a requisite for scripta with multiple versions of the same work. We have some good URI vocabulary for famous works, and we have some good URI vocabulary for items of literature where the work-scriptum divide is not so strong (modern scholarly articles). But how do we find vocabularies of work-versions that aren't famous and are clearly down the version path? What about work-versions that are based on arbitrary categories? The previous two paragraphs note that work-versions can be built in various combinations, either through the genetic relationships that make up stemmatology or seemingly arbitrary properties.

The problem is somewhat philosophical. A work is ultimately a class of abstract, conceptual objects. Any restriction of that class, into a work-version, requires the stipulation of acceptable values for one or more properties. Such properties can come in a bewildering variety of types. The point seems abstract now, but it has real importance when it comes to qualifying a particular work-based URI. When pointing to particular version of a work in a book (say, facing text and translation) how does on restrict the scope of a work URI to that particular version?

Scripta

[Definition: A *scriptum* is a physical item of writing. It is a text-bearing material object, or a set of (largely) indistinguishable copies of text-bearing objects. Every scriptum is a physical entity with shared limits, whose content admits division (perhaps into other scripta), and whose content cannot be equated or conflated with any single work.] For the purposes of WF URIs, digital files are to be regarded as material objects, and therefore scripta. Manuscript Venetus A, Caroline Alexander's translation of the *Iliad* (ISBN 9780062046277), the 1594 first quarto edition of *Henry VI Part 2*, the Bate and Rasmussen edition of *Henry VI* (ISBN 9780812969405), and a pdf scan of that edition, are all examples of scripta.

A digital scan or digital photograph of a material object *must not* be treated as the same scriptum as what it represents. It is a digital surrogate, and constitutes a distinct scriptum.

A scriptum may be a constituent part of another scriptum (e.g., a volume in a multivolume publication) or it may comprise multiple scripta (e.g., a volume that consists of multiple fascicles). Sometimes a single scriptum may be legitimately treated as overlapping or intersecting another (e.g., a manuscript that was broken apart long ago, the parts joined to other manuscripts). It may be possible for a single artefact to be treated wholly as two different scripta. For example, a hand-annotated print copy of a book may be considered an individual manuscript (a scriptum with set membership of one) or as part of a print run (a scriptum with set membersip of more than one).

Shared Limits

[Definition: The the term *shared limits*, used of works and scripta, refers to a consensus held by a community of practice.] [Definition: A *shared work limit* is a consensus by a community of practice on what the limits are to a particular work.] [Definition: A *shared scriptum limit* is a consensus on the limits for a particular scriptum.] In many cases these distinctions are not significant, because most such limits are shared not simply by a community of practice, but by everyone.

In the case of the shared limits of scripta, WF does not define "indistinguishable," the criterion used to determine whether any two material objects are the same scriptum. If the limits of a scriptum are unclear, it should be avoided as the basis of a WF URI.

In the case of some works, especially culturally influential works, one community's shared limits will differ from those of another (e.g., the limits to the meaning of "Bible"). The question of resolving ambiguous or contested shared limits is discussed below.

The Relationship between Works and Scripta

The description of works and scripta above resembles the account of Type 1 Entities defined by the Functional Requirements for Bibliographic Records (FRBR) [https://en.wikipedia.org/wiki/Functional_Requirements_for_Bibliographic_Records]. But whereas FRBR Type 1 Entities distiguish between Works and Expressions, the WF model treats all conceptual entities (i.e., both Works and Expressions) as a single category. Any rendition or version of a conceptual, non-material work is itself a conceptual, non-material work, even if it closely depends upon another. The FRBR term *Expression*, said to refer to a nonmaterial, conceptual entity, is avoided because no one can express a work without some material medium through which to express it. Similarly, whereas FRBR distinguishes between Manifestations and Items, the WF model combines both, applying *scripta* to all material text-bearing objects. If scripta are mechanically reproduced and are largely indistinguishable from one another, they are treated as the same scriptum. Thus, WF *scriptum* most resembles FRBR *manifestation*, which is applied to sets of material items.

Any single work may be realized in one or more scripta. A work may be represented in a scriptum completely, partially, or repeatedly. Likewise, any given scriptum will contain at least one works. Each work is represented completely, partially, or repeatedly. Many types of scripta have multiple works (e.g., an anthology; a compilation; or a bilingual edition, which has two independent work-versions). The work or works on a scriptum may not be identifiable (e.g., a papyrus fragment with unattested text). Some works do not survive in any known scripta (e.g., Aristotle, *Poetics* book 2, on comedy; the letters written by the Corinthians to Paul the Apostle; George Orwell's *Socialism and War*).

In many cases, particularly in ancient and medieval literature, the distinction between works and scripta will be readily apparent. In other cases, the distinction is not always evident. For example, many scholarly articles are assigned a Digital Object Identifier (DOI) URI, which may be assigned by the publisher to the item *qua* work, *qua* scriptum, or both. Others may use that same DOI URI to refer to the article as a work, as a scriptum, or as both. Anyone who creates a WF URI must therefore specify whether a given URI refers to a writing *qua* work or *qua* scriptum.

Scripta: Readers, Regions, and Text Structures

Given the framework adopted above, several more key concepts follow, centered around those who use scripta. WF URIs have been designed under the assumption that anyone who independently approaches a scriptum endeavors to understand the shared features of that scriptum. [Definition: The *shared features* of a scriptum include its work-version regions (defined below) and their parts.] No WF URI may be applied to a scriptum that does not lend itself to such shared features.

[Definition: The term *scriptum reader* is defined as a person who has the aptitude to understand the text in a given scriptum, either because that person is part of the audience intended by the creator of the scriptum, or because they have acquired the background needed to have that facility.] A scriptum reader can read at least one of the written languages in the scriptum, and has the requisite background to understand what the scriptum is about, and to discern the scriptum's most significant shared features.

Warning

From this point, the argument stresses the competence of what is termed the *scriptum reader*. It pivots away from defining textual objects *an sich* to defining the perceiver of the textual object. Is this a legitimate approach? If so, what are the implications (epistemological esp.)? If not, what are the alternatives? It seems you want as many people involved in making WF URIs, but you don't want dilettantes and fools. But perhaps even the most competent of us might think the other a dilettante and fool in some respect?

[Definition: A *work-version region* is defined as a set of one or more places on a scriptum that collectively delimit all the text that is part of a single version of a single work.] For example, a bilingual

novel lends itself to several different work-version regions: the original text, the facing translation, the prefatory introduction, an excursus in the appendix, etc. A work-version region need not be contiguous. For example, a scriptum may feature scattered quotes from another text. Work-version regions, like work-versions themselves, may nest, overlap, or tesselate.

Most work-version regions are designed to help scriptum readers distinguish clearly between [Definition: the *main text*, the text that is squarely part of the work itself and within its shared limits], and [Definition: the *paratext*, features intended to divide, structure, and order the main text.]

Most main texts can be distinguished by scriptum readers into a [Definition: single *trunk text*, a text that runs from start to finish (logical, perhaps also physical),] and zero or more [Definition: *branch texts*, texts that are anchored to a point or range in the trunk text, and that relate to the trunk text at or near the anchored region]. Examples of branch texts include footnotes, endnotes, glosses, and marginalia.

Paratext includes not only visible components that can be isolated from the main text, such as symbols and spaces (margins, indentations, leading, etc.), but also recognizable attributes of the main text (color, font, size, weight, etc.) that a scriptum reader would recognize as organizing the main text. It may the case that paratext is interspersed within the main text (e.g., vertical bars to signify line breaks, inline numerals to mark the beginning of a textual unit). Any visible features associated with the work that are unclear in function (e.g., marginal doodles) are to be treated as paratext.

[Definition: A *viable work-version region* is a work-version region whose main text, paratext, trunk text, and branch texts are clear to scriptum readers.]

Viable work-version regions can be named, whether by traditional nomenclature or URI. For example, in the bilingual edition example above, the viable work-version region for the version in the original language can be named with URIs reserved for the main work. It can also be given URIs specific to the particular work-version (i.e., X's edition of the original text). The same can be said of the facing translation: it can be named after the general work itself, and after the specific work-version.

Warning

Note the discussion above about the difficulty of restricting a work to a work-version. I see different ways to approach the task of building a URI for a work-version:

1. **Absolute base URI**. Each work-version must take only absolute URIs. The relationship between work-versions should not be built into the URI architecture itself. If anyone wishes to state the relationship held by any two work-versions, that would need to happen separately, in RDF triples.

   Advantages: everything is rooted in URIs, without a second level of fragids to memorize.

   Disadvantages: there hardly exist any such absolute, specific work-version URIs. No one is entitled to change a URI in a namespace they don't own (absent URI fragments), so they would have to be coined by individual users in their own namespaces. Wouldn't this result in numerous obscure vocabularies? (Elsewhere I've admonished creators of work URIs to provide a mechanism whereby they can mint sub URIs for specific work-versions, with the hope that maybe Wikipedia could find a way, perhaps with #-identifiable stubs. Would these be okay?)

2. **Work fragids**. The idea here is to define, separately, another fragid structure exclusively for work URIs. One can then insert parameters that restrict the class along any number of criteria. For example, `<http://dbpedia.org/resource/Iliad#$wf0:lang=grc$>` would declare restriction in scope to only works of the Iliad in the Greek language.

   Advantages: it's modular, and can allow for a variety of properties to restrict classes. It allows for a common set of work URIs that don't need to be guessed at.

   Disadvantages: permutations would abound. ISO 639-3 language codes are easy, but what about defining authorship or date or place of creation? One would expect, given a new fragid structure, that the parameters would need to refer to URIs and controlled vocabulary, not nick-

names. It could get c-r-a-z-y in length and legibility. WFs would have to specify what classes are permitted. But then how does one know whether a particular media file that matches the base work URI also matches the subclass?

3. **Adapted URIs**. This would be an attempt to thread the needle between the previous two options. A user would embed the best known Ur-work URI within a tag URI in the user's namespace, then add in specific WF-defined steps to restrict membership, e.g., `<tag:example.com,2014:http://dbpedia.org/resource/Iliad/**2002/grc/west**>` would identify West's specific version, but `<tag:example.com,2014:http://dbpedia.org/resource/Iliad/**\*/grc/\***>` would point to any Greek edition.

   Advantages: the tag uri attaches provenance to the new URI for blame/credit. The extra paths lend themselves to shorthand nomenclature that could attract matches based on shared practices. Only a few of the most common examples would be permitted, to avoid chaos.

   Disadvantages: The URI is no longer opaque: processors would have to drop the authority part of the tag URI before trying to make matches. What do you do with fuzzy dates? Or ones where lots of creators would be inserted in the third step. Everyone will complain: you don't support X as a criteria. Even supported work-versions could result in wild and crazy unpredictable constructions. It is also built upon not on the work-version path, but arbitrary criteria meant to imitate the work-version path.

Perhaps there's another way forward? We need some sort of convenient way to say something like "West's version of the Iliad" (regardless if it's the original Greek or a modern translation) or "all versions of the Iliad published in the 1950s." It doesn't mean WF needs to support all possible permutations, but it should have some kind of controlled template that would permit growth.

[Definition: A *work-scriptum map* is defined as scriptum allocated into only those viable work-version regions that admit locally unique URIs.] The work-scriptum map may be described procedurally. Apply work-version regions to a scriptum. Remove any that are not viable. Apply to each remaining region all possible URIs that identify its work-version or any inherited work-versions (see above on work-version inheritance). Find all regions with duplicate URIs. If one region contains a work-version clearly older than the rest (it has the shortest version path to a putative original), let it uniquely retain that URI, and discard all other instances of the duplicate URI. Discard any regions that do not have any URIs.

For example, a bilingual French-Greek edition of Aristotle's *Categories* will admit at least two work-version regions. Because the French is a translation of the facing Greek, and depends on it, only the Greek version retains the URI for the work in general, as well as the URI for that particular version. The French version retains only the URI for the French version.

The work-scriptum map and its work-version regions are important, because they allow one to unambiguously identify select parts of a scriptum based on works, and assign priority to multiple versions of the same work. If a particular work-version features two or more times in a scriptum, and one of them does not clearly have priority (the shortest version path), the general work URI will *never* be used, only the URIs that are distinct to unique work-versions.

The Trunk Text: Divisions, Units, Sequences, and Trees

In this section "the text," when not specified, refers to the main, trunk text of a work-version region in a work scriptum map. It excludes the paratext (e.g., page numbers) or branch texts (e.g., notes).

Definitions pertaining to divisions and units focus on scripta and exclude work-versions per se, because two people can discuss a text's divisions and units only on the basis of real-world examples. Works are abstract entities that may be conceptualized differently (one person's mental work divisions cannot be consulted by another), so cannot define text divisions and units. One needs scripta.

[Definition: A *textual division* marks a break in the text.] Examples of textual divisions in scripta include spaces (indentations, margins, word spaces), labels, hairlines, and changes in text rendering (color, font, weight, size). Textual divisions are to be treated as anchors within a stream of text, and always intervening between text items (characters).

[Definition: A *textual division group* is a series of textual divisions that divide a text, or a designated portion of a text, by the same units with a thematically related set of labels.]

[Definition: Applying one textual division group to a text, or to a designated portion of a text, results in a sequence of one or more *textual units*.] Textual units are normally named in terms of division typology. Examples of textual units include books, pages, columns, lines, books, chapters, parts, stanzas, periods, indentations, and word spaces.

Any textual unit might itself be subject to division by another textual division group and result in a subordinate textual unit sequence. A page, for example, might be divided into lines. A textual unit sequence might be composed of textual units of different types. For example, a chapter might include a mixture of paragraphs and run-in headers.

Most textual unit sequences will be ordered according to a single directed stream (a main text consisting of a trunk text and perhaps branch texts). This merely reflects the nature of the divided text. Some textual unit sequences may take multiple streams, or be undirected or semi-directed (e.g., concrete poetry, word clouds). Regardless of how a text is ordered or directed, its textual unit sequence will inherit that order and direction.

Although rooted in scripta, a given textual division or textual unit might be defined by an ideal, not a material entity, and therefore be grounded notionally in works, not scripta. All textual divisions are based either on the *physical, material features of a scriptum*, or upon *logical, conceptual contours of a work*. Examples of material textual units include pages, lines, columns, subcolumns, codexes, fascicles, and folios. Examples of logical textual units include abstracts, acts, appendixes, chapters, suras, paragraphs, couplets, stanzas, sentences, notes, quotations, verses, phrases, words, and letters.

The number of logical unit types are significantly greater than material unit types. They are also generally not as well defined, because they label conceptual entities, not material ones.

Some textual unit names can be interpreted as being either material or logical, e.g., books, volumes, parts. Context determines whether such a term refers to a material or a logical textual unit, oftentimes indicated by the next unambiguous subdivision unit type. If a book is divided into chapters, the book unit is logical; if it is divided into pages, it is material.

[Definition: A *textual unit tree* is a text divided into a single sequence of textual units (its primary sequence), any unit of which may itself be divided into a subsequence of textual units, and so forth.] The tree metaphor has further application. Any given textual unit either yields sequences of smaller ones, and therefore can be called a branch, or it terminates, and is so called a leaf.

Every viable work-version region in a scriptum can be organized into zero or more textual unit trees. For example, a book (the entire scriptum being taken as the viable work-version region) might have a textual unit tree based on pages, columns, and lines. That same book may have another textual unit tree of parts, chapters, subchapters, paragraphs, and sentences. That same book may include yet other textual unit trees (e.g., a second level of pagination, corresponding to a previous edition). In any given textual unit tree, no textual unit is to be divided into more than one sequence. If a textual unit permits an alternative, second sequence, it must be expressed in a separate tree.

A textual unit tree can be of any depth. No tree is required to have a depth greater than one. Consequently, every textual unit sequence is also a textual unit tree.

Note

The WF model differs from the common ideal of a text as an ordered hierarchy of content objects (OHCO), coined by Renear et al. 1990. The WF model does not insist that text is ordered (the O in OHCO), but this version of the WF syntax supports only ordered, directed text. (Unordered,

undirected text may be supported in a future version of WF.) Hierarchy (the H in OHCO) is reframed in terms of textual unit sequences, which can nest (but are not required to do so) to produce hierarchies. The terms "content" and "objects" (the CO in OHCO) are regarded as redundant. An ordered textual hierarchy cannot have non-content or non-objects.

Every textual unit sequence is classified as being either *material* or *logical* depending upon the criteria for the division group. A textual unit tree may therefore also be termed material or logical, based on the classification of its primary (initial) textual unit sequence. For example, a page-column-line textual unit tree is material, whereas a part-chapter-subchapter-paragraph-sentence tree is logical.

A textual unit tree may be either native or adopted. [Definition: A *native textual unit tree* is one that has been created for the particular scriptum.] [Definition: An *adopted textual unit tree* is one that has been applied to a scriptum from a previous one.]

Some viable work-version regions may allow many concurrent textual unit trees, both material and logical, native and adopted. For example, a modern translation of Aristotle's *Categories* may have (1) a unique sequence of paragraphs (a tree of native logical textual units); (2) page and line breaks (native material units); (3) sectioning found in many other editions and translations (adopted logical units); and (4) line numbers drawn from Bekker's 19th-century edition (adopted material units).

Labels; Reference Units, Sequences, and Systems

In a textual unit sequence each unit may be labeled or not. [Definition: A textual unit is *labeled* if the unit is accompanied by numerals, letters, abbreviations, or symbols in the paratext that names or identifies the unit relative to the others in the same sequence.] Such labels provide the explicit basis for reference systems.

A label does not necessarily identify a clearly demarcated textual unit. Some labels identify textual units that are not clearly divided, or not divided at all. For example, a modern translation of a text by Plato may have occasional Stephanus numbers (the page numbers of a 19th-century edition of the Greek) in the margin, but the paratext might also lack any clear textual divisions to specify where one textual unit ends and the other begins. Such vague labels tend to come with trees based on adopted material units. From this point forward, the term *label* excludes these types of vague labels, and means only those labels that are attached to textual units with clear divisions.

Any textual unit sequence is, on the whole, a labeled sequence or an unlabeled one. [Definition: A *labeled textual unit sequence* is one where the majority of textual units are labeled.] [Definition: All others are termed *unlabeled textual unit sequences*.]

Within a given labeled textual unit sequence every textual unit label is unique or non-unique. [Definition: Any label attached to only one textual unit in the sequence is a *unique label*. Any label attached to two or more textual units (i.e., two or more units with the same label) must be treated as unique if the label, according to shared convention, marks the textual units as component parts of a single range of text.] [Definition: All others are *non-unique labels*.] For example, in the label sequence 1, 2a, 5, 2b, rubric, 3, [unlabeled], 4, rubric, 6, 7... the label "rubric" is non-unique, and all others are unique. Scriptum readers can discern from the pattern that the labels "2a" and "2b" form a pair that designate the first and second half of a single range of text, so are treated as jointly forming the unique label "2." [Definition: Any textual units joined by a single label, whether the units are contiguous or not in the sequence, are called *reference units*.] Thus, the two textual units marked by 2a and 2b in the previous example are the two parts of a single reference unit.

In a textual unit sequence some or all of the unique labels may discerned to be part of one or more ordered systems. [Definition: An *ordered label system* is a set of unique labels that have a common syntax drawn from a shared ordered system rooted outside the particular work or scriptum.] An example of an ordered label system would be "A, B, C, D." But "Matthew, Mark, Luke, John" would not be an ordered label system, because the syntax and order are unique to a single work, the New Testament (and that order is not universally shared).

Any ordered label system (e.g., Roman numerals, Indic numerals, letter labels, enumerated dot clusters) can be converted to a sequence of integers, or integer-modified integers. [Definition: *Integer-mod-*

ified integers are integers with subintegers to specify a subsequence.] Integer-modified integers accommodate supplemental reference units. For example, 1, 2, 2a, 2b, 3 would be converted to the following sequence of integers and integer-modified integers: 1, 2-0, 2-1, 2-2, 3.

The logical sequence of an ordered label system may differ from the label order in the scriptum. This commonly happens in editions of versions of texts that depart from the normal sequence. Oftentimes the editor places the traditional reference labels by text order, to align it with the customary reference system.

[Definition: A *uniquely ordered reference unit sequence* is one that has only one ordered label system.] Such a sequence may include interleaved textual units that are unlabeled, or labeled but unordered. For example, a sequence with the labels "title, subtitle, 1, 2, 3, [unlabeled unit], 4, 5" would be a uniquely ordered textual unit sequence. But the sequence "1, 2, a, b, c, 3, 4" would not, because it has more than one ordered label system.

The previous several paragraphs are relevant only for labeled textual unit sequences. [Definition: An unlabeled textual unit sequence may lend itself to an *implicit unique ordered reference sequence*, i.e., one where the textual units, although not explicitly labeled, can be unambiguously enumerated by scriptum readers,] [Definition: and supplied *implicit labels*.] For example, most books published today do not label line numbers. But lines of main text may be unambiguously enumerated by anyone to create and use an implicit unique ordered reference system, e.g., page 2, line 3. Similar consideration may be extended to cases where a textual unit sequence has periodic labels that infer an ordered label system. For example, a page with line numeration only on every fifth line may be reliably assigned implicit labels for every line.

WF URIs can rely upon implicit unique ordered reference systems, but only on the basis of consistent unit type classes. No logical unit may be subdivided into implicit material units, or vice versa. This principle avoids the possibility of incomplete textual units and ambiguity or confusion in implicit numeration. For example, a sequence of pages (material textual units) may not be subdivided into paragraphs (logical units), even if no paragraph in the scriptum in question is broken by pages. On a similar principle, lines (material units) be not be subdivided into words (logical units).

A special exception is made for enumerated branch texts such as endnotes or footnotes (logical units), which are traditionally referred to in concert with the page number (material) on which the note starts. In such cases, a note (a branch text) is considered a subdivision of a page.

In WF URIs, *no implicit ordered reference system may be applied to textual units that do not have clearly shared limits*. Most often this happens with logical textual units. Paragraphs, sentences, and words are frequently difficult to enumerate because many of them do not have clear shared limits. For example, a paragraph may be interrupted by a block quote that itself contains multiple paragraphs. Some readers may consider the resumption of the original paragraph after the block quote to be a new paragraph in its own right. Some readers may wish to include the block quote's paragraphs as part of the implicit enumeration. A similar problem occurs with sentences, which easily nest, e.g., "I said, 'Stop. Wait. Listen,' before the clock chimed." This proviso applies only to unlabeled textual units. If paragraphs, sentences, or words are given ordered explicit labels, they may be used.

[Definition: A *reference system* consists of the explicit and implicit labels of a textual unit tree, preserved in their treelike structure.] A reference system may include both ordered and unordered labels.

[Definition: A reference system is navigated through coordinates called *references*.] References may be simple or compound. [Definition: A *simple reference* consists of a sequence of labels that points step by step to one reference unit in the tree, to provide instructions on traversal from the root of a textual unit tree toward its leaves.] For example, the following four simple references (1), (1, a), (1, b), and (2) would point to the four leaf textual units in the following reference tree: (1 (a, b), 2).

Note

In this and subsequent examples, different numeration systems (Arabic, Roman, alphabetic) are used to clearly indicate levels within the reference tree.

[Definition: A *compound reference* is an ordered sequence of one or more simple references.] For example, (1-2) or (1-3, (6, a), 9). A compound reference is independently ordered. It need not follow either the order of the reference system, or the order of the reference units.

A reference system may include a variety of ordered and unordered label systems, and it may include non-unique labels. A reference system will correspond one-to-one with every node in a textual unit tree except for those textual units that are unlabeled and do not permit implicit labels.

Special types of reference systems may be created by restricting reference systems based on certain criteria. All WFs rely on a special class of reference system termed a *unique, ordered reference system*. [Definition: Any reference system may be converted to a *unique, ordered reference system* through the following steps:]

1. *Exclude any branch of the tree that is not a member of a uniquely ordered reference system*. The process may described recursively. At the first level of the reference system, only uniquely ordered reference units are retained, as well as their uniquely ordered reference units, and so on, with the process stopping at the leafmost parts of the tree, or at units whose subdivisions do not have uniquely ordered reference units. For example, a tree with a reference system such as (title, 1 (a, b), 2, [unlabeled], epilogue) sheds three labels and their branches, and is reduced to a subtree with a unique, ordered reference system: (1 (a, b), 2).

2. *Skip any otiose levels*. In a textual unit with a uniquely ordered reference unit sequence, if there are one or more deeper levels that aggregately constitute an unique ordered reference unit sequence for the entire textual unit, then the reference tree should skip to that deepest level. For example, a book might be divided into sequentially numbered parts, and then into numbered chapters that do not restart numeration with each new part. In such a case, the numbered parts are otiose: the chapter numbers are the deepest first unique ordered reference unit sequence for the entire book. Another example: some Greek Bibles divide the Psalms into twenty *kathismata* ("sittings") and then into one hundred fifty sequentially numbered Psalms. The numerals 1-150 are the deepest reference unit sequence, and the otiose *kathisma* numbers should be ignored.

   Such otiose sequences may appear not merely on the top level, but internally, which is why the definition is expressed in terms of a textual unit. For example, the reference system tree (1 (A (i, ii), B (iii, iv)), 2, 3) must under this rule be reduced to the following unique ordered reference system: (1 (i, ii, iii, iv), 2, 3). Such a principle is applied level by level, and pertains only to label systems at the same depth within the tree. The following reference system tree cannot be reduced under this rule: (1 (A (i, ii), B (iii), C (a (iv, v)))), 2, 3).

A reference system may be classified as material or logical, based upon the type of textual unit of the primary textual sequence. For example, a reference system built on page then line numbers would be classified as a material reference system because pages are material units. One built on book-chapter-sections (many modern editions of ancient literature are so organized) would be a logical reference system because "book" here is a logical unit (signaled by the logical nature of the next subdivision, chapter), not a physical one.

[Definition: A scriptum has a *key material reference system* if and only if it has only *one unique, ordered material reference system, native to the scriptum*.] Many printed articles have only one set of material references: page numbers, and implicit line numeration and explicit note numeration. Such articles have a key material reference system. But if one of those articles is reprinted in a monograph, with pages counterstamped with the book's pagination, then that particular scriptum lacks any key material reference system. If a book has a native reference system based on its own page and line numbers, but it also incorporates reference systems based on manuscript folios or page numbers from previous editions, it still has a key material reference system, because this definition excludes from consideration adopted material reference systems.

[Definition: Similarly, a scriptum has a *key logical reference system* if and only if it has only *one unique, ordered logical reference system*.] Many books have enumerated or enumerable parts, chapters, and sections. Such books have a key logical reference system. Many legal codes, regulations, or other texts organized as an enumerated hierarchy have a key logical reference system. Editions of

the Bible and Qur'an that rely upon a unique division of chapters and verses or suras and ayahs each also have a key logical reference system. But this is not the case for a scriptum that applies multiple overlapping logical reference systems, even if one is given greater prominence than the others.

The above definitions for key reference systems are stipulated for a viable work-version region that is coextensive with the scriptum taken as a whole. The definitions are slightly relaxed for a scriptum restricted in scope to a particular viable work-version region, [Definition: termed a *work-version-constrained scriptum*, or simply a *constrained scriptum*.] For example, a book of fairy tales contains numerous works. One may constrain the scope of the scriptum to one single version of one fairy tale before testing for a key logical reference system.

In a constrained scriptum, the material reference system may be either native or adopted. For example, if an article reproduces significant portions of Aristotle's, *Categories*, and those portions are labeled with line numeration based upon a previous edition, and there are no other material reference systems (e.g., references to folio and line numbers from manuscripts), then the adopted material reference tree is the key material reference system.

Similarly, in a constrained scriptum, the key logical reference system is assessed in light of both native and adopted reference trees. If a constrained scriptum, in the form of a poem, is given stanza and (logical) line numbers based on prior editions, and there are no other logical reference systems present, then the imported reference system may be treated as the key logical reference system.

[Definition: A scriptum or a constrained scriptum is said to be *WF-viable* if and only if it has *one key material reference system, one key logical reference system, or both*.] [Definition: Otherwise the scriptum or constrained scriptum is termed *WF-non-viable*.]

General Stipulations

Important

This section is normative.

From the challenges, concepts, and definitions detailed in the previous sections, some general strictures on Writing Fragids follow. Syntactic requirements are reserved for the next section.

Every WF must declare whether the base URI points to a work or to a scriptum. If the latter is the case, the base URI may be converted to a constrained scriptum by supplying a second URI that specifies the work-version constraint. There are, then, a total of three possible types of WFs, termed *work WF*, *scriptum WF*, and *constrained scriptum WF*.

Warning

To reduce the amount of permutations, I have not supported **constrained work WF*, that is, one that begins with a work-version URI at the base, then qualifies it with a scriptum. My thought is that the difference between a constrained work WF and a constrained scriptum WF are negligible. But I have doubts.

Every WF must include a URI that points to [Definition: a *reference scriptum*, a WF-viable scriptum that serves as the basis for the reference system]. Every WF must declare whether it is reliant upon that reference scriptum's key material or logical reference system.

Any WF may point to any node within the declared key reference system, whether a branch or a leaf, via a simple or compound reference.

Any scriptum WF or constrained scriptum WF may add to a simple reference a further pointer to a text fragment in a reference unit (phrases, words, characters). But this does not apply to a work WF. A work WF points to every version of a work, so has scope over potentially numerous languages and scripts. Because no reference unit corresponds to a single text, a text fragment may not be added to a simple reference.

Warning

If the approach to work-version URI construction changes, this stricture might need to be reconsidered.

The WF model does not support non-enumerated references, e.g., Gen. 1:1 for the first verse of Genesis, because the label "Gen." is not ordered. In this case, Genesis is a work in its own right. The base URI should point to Genesis *qua* work, and the WF portion to 1:1.

The WF model does not support references that rely upon vague implicit subdivision, e.g., 1:1b, when the scriptum that forms the basis for the reference system does not clearly subdivide 1:1 with labels, and an implicit unique ordered reference sequence is not evident. Subdivisions may be used only when the subdivisions are explicitly labeled.

A single WF URI may be given more than one reference. In such cases, the URI stipulates an ordered sequence. That is, a WF URI pointing to references 1, 5, and 2 entails reference 1 then 5 then 2, treated as a single entity.

Any WF URI reference that points to only one text (i.e., a scriptum WF or a constrained scriptum WF) may be restricted further in scope, to a particular range of text (whether a single character, several characters, a token, a phrase, or a longer span of text).

The previous two stipulations anticipate cases where the writer of a WF URI wishes to point to ordered passages and phrases, perhaps to declare where one text quotes another, and to do so respecting the order in which the source quotes the target.

There is another anticipated use case, where the writer of the WF URI wishes to point to individual linguistic units, i.e., tokens or token fragments. To point to individual tokens must create one WF URI per token. A future version of the WF specifications may support URIs that point to multiple instances of a token in a target text.

Warning

I have adopted this position only after first trying a more flexible syntax. But I got confused quite a lot when I started to think about what the URI is supposed to designate, particularly when a single WF URI pointed to a mixture of token-based text fragments and phrase-based ones. Further, the instance selector had to be opened up to multiple integers. A requirement that integers appear in ascending sequence could not be defined in the syntax specs. In the end, I felt that a URI should be designed to point to a single entity.

Restrictions above apply to this version of the WF guidelines. Stipulations may be added, changed, or removed in future versions.

Syntax

Important

This section is normative.

Warning

Forward-compatibility: Because this is a draft specification, no guarantee is made that the syntax defined below will be compatible with the syntax defined in future specifications.

Base URI

Every WF URI consists of a URI whose fragment section includes a WF. A WF may be combined with other fragment sections. The base URI must conform to the syntax for URIs defined by RFC 3986

[https://www.ietf.org/rfc/rfc3986.txt], whose requirements are not repeated here, except for relevant fragments.

The URI is defined as follows (*3. Syntax Components*):

```
URI ::= scheme ":" hier-part [ "?" query ] [ "#" fragment ]
```

Warning

Are there any possible conflicts between a query constructor and a WF fragment? Do LOD practitioners discourage queries?

RFC 3986's definition of `fragment` at *3.5. Fragment* is extended, to explain WF syntax more precisely (expressed with slight adjustments in syntax):

```
fragment ::= stdFragment* LitFrag stdFragment*
```

RFC 3986's definition of `stdFragment` is as follows (reproduced unchanged for contextual reference):

```
stdFragment ::= ( pchar | "/" | "?" )*
pchar ::= unreserved | pct-encoded | sub-delims | ":" | "@"
unreserved ::= ALPHA | DIGIT | "-" | "." | "_" | "~"
pct-encoded ::= "%" HEXDIG HEXDIG
sub-delims  ::= "!" | "$" | "&" | "'" | "(" | ")" | "*" | "+" | "," | ";" | "="
```

WF adds the following supplemental definitions:

```
LitFragUri ::= LitFragUriChar+
LitFragUriChar ::= LitFragUriPChar | "/" | "?"
LitFragUriPChar ::= unreserved | pct-encoded | LitFragUriSubDelims | EscapedData
LitFragUriSubDelims ::= "!" | "'" | "&" | "(" | ")" | "*" | "+" | "," | "="
EscapedData ::= "^$" | "^;" | "^^"

LitFragUnicodeChar ::= [#x0-#x10FFFF] - [$^\[ :-#] | "^$" | "^[" | "^:" | "^-"

INTEGER ::= ( DIGIT - "0" ) DIGIT*
```

A WF is delimited by its beginning and ending:

```
LitFrag ::= LitFragStart LitFragBody LitFragEnd
LitFragStart ::= "$lf0:"
LitFragEnd ::= "$"
```

The opening line specifies the version of WF (major version only). The WF does not end until the unescaped `LitFragEnd` is encountered.

Many linked open data vocabularies make use of `https?://` style URIs with a fragment section that points to a value of `@id` in the target web resource to name something. In those cases, the first string in the fragment should be treated as an inseparable component of the base URI. The fragment is an essential part of the URI that names the resource.

The syntax formally defined here does not preclude multiple WFs in a single URI. It is nevertheless disallowed. A URI should be treated as invalid if it is modified by more than one WF.

The body of a WF consists of two parts: parameters and references.

Parameters

Narrative

Several parameters are required in a WF, to declare the nature of the URI and the reference system that has been used. Parameters come in the form of key-value pairs, joined by an equals sign and delimited by semicolons.

There are three types of WF URIs: work WFs, scriptum WFs, and constrained scriptum WFs. The first is designated by the first parameter `a=w;` (syntax adopted from RDF conventions, where `a` typically means "is a"). The second and third types take `a=s;`.

In a constrained scriptum WF there follows the parameter `w=` followed by a URI pointing to a work. Any special characters must be escaped; it terminates in an unescaped `;`.

All three types of WF URI must then specify the type of reference system used: `t=m;` (for "type: material") or `t=l;` ("type: logical").

There then follows a parameter declaring the reference scriptum URI, which begins `r=` followed by a URI that points to a scriptum. All special characters must be escaped; it terminates in an unescaped `;`. The reference scriptum is the basis of the WF URI's key reference system. A work WF *must* supply a reference scriptum URI in full, unless the base URI in a work WF can be interpreted doubly as a scriptum and a work, in which case `r=.;` specifies that the base URI is identical to the reference URI, interpreted *qua* scriptum. A scriptum WF or a constrained scriptum WF may simply state `r=.;` if the base URI scriptum is also the reference scriptum, otherwise the reference scriptum URI must be supplied in full.

The final section in the WF are the references, described in the next section.

As defined by RFC 3986 and these specifications, which forbid # from appearing in any URI fragment, any WF parameter that takes a URI with a fragment *must* percent-escape the # (`%23`). This process may require recursion, since that fragment might itself be a WF URI with a percent-escaped # (`%23` → `%2523`).

Defined Syntax

```
LitFragBody ::= WorkLitFrag | ScriptumLitFrag | ConstrainedScriptumLitFrag
WorkLitFrag ::= "a=w;" RefSystemType RefSystemBasis ReferencesOnly
ScriptumLitFrag ::= "a=s;" RefSystemType RefSystemBasis ReferencesWithText
ConstrainedScriptumLitFrag ::= "a=s;" WorkConstraint RefSystemType RefSystemBas
WorkConstraint ::= "w=" URI ";"
RefSystemType ::= "t=" ( "m" | "l" ) ";"
RefSystemBasis ::= "r=" ( "." | URI ) ";"
```

Conformance

The designer of a WF-media type *must* define in their guidelines how any file in the format can be tested for a content match against all the parameters in a WF: the base URI, a scriptum-constraining work URI, the reference system type, and the reference scriptum URI (either in full or with an `.` to point to the base URI).

Authors of WF-conformant formats *may* define a format-specific mechanism to convert any URI for a work, a scriptum, a scriptum-constraining work, or a reference scriptum into one or more synonymous values, or it *may* define a mechanism to allow a data creator to do so. Whether a WF processor takes advantage of those mechanisms is implementation-dependent.

Every designer of a WF-conformant media format *must* allow a data creator to assign to a file or designated file fragment each of the following:

1. One or more URIs identifying the work;
2. One or more URIs identifying the underlying scriptum;
3. Assignment of textual unit labels corresponding to a reference system tree.
4. For each reference system used:
    a. A value specifying the reference system type: material or logical;
    b. One or more URIs identifying the WF-viable reference scriptum;

The above components not governed by rules for URIs *may not* be case-sensitive. Every WF-conformant media format *may* allow multiple reference systems in the same file, but *must* define a mechanism that allows a processor to extract from the content a tree consisting exclusively of the target key reference system.

A processor *must not* process a WF URI that does not conform to the syntax defined above.

Processors of WF-conformant data *must not* match a WF URI against a file or file fragment where there is not a match (whether exact or a derived alias) in each of the major components (work, scriptum, reference system scriptum) declared in a work WF, a scriptum WF, or a constrained scriptum WF.

A WF processor *may* flag as erroneous any URI portion of a WF URI that does not conform to the constraints established by the declared scheme (the first part of any URN). For example, an http URI may be checked for path validity. A URI starting `urn:` may be checked against the IANA registry [https://www.iana.org/assignments/urn-namespaces/urn-namespaces.xhtml] to see if the declared URN authority is officially recognized. A tag urn may be checked for namespace syntactic validity. The types of checks, their depth, and the response to any errors, are implementation dependent.

Parts of WF parameters not governed by rules for URIs are case-insensitive. Case rules for URI components follow the rules specified in RFC 3986. A WF-processor *must* normalize any case-insensitive WF components to upper- or lowercase before parsing a WF URI and comparing it to counterpart values in WF-conformant media.

References

Narrative

The last half of the WF body consists of a sequence of references, joined by &, and treated as a single entity. All references are ordered. If two WF URIs are the same except that one has references A then B and the other has B then A, then they are not identical.

A reference (a member in a sequence of references) consists either of a reference unit or of a reference range. A reference range is a stretch of text delimited by two reference units. Although the first reference unit in a reference range will normally precede the second, this is not required. The reference units in a reference range are joined by a hyphen.

A reference unit consists of a sequence of reference steps, delimited by colons. Each reference step consists of either an integer or two integers joined by a period. If a footnote or endnote is intended, the first integer must be prefaced by `n`.

Warning

A note (footnote, endnote, etc.) can participate in either logical or material systems, but most often material. And material systems have far fewer division types than physical division sequences. It is possible to predefine the set of permitted material textual units, and assign a specific code: v for volume (nestable), p for page, c for column, sc for subcolumn, f for folio, s for side, b for block (newspapers), z for line (the letter ell will always look like a numeral 1). (There are a few other

material unit types I can think of, but they are either edge cases, e.g., quire/gathering/signature, or lend themselves to inclusion with the other categories, e.g., fascicle, codex, volumen.)

Logical units, on the other hand, are in great abundance, and cannot be succinctly typed with a prefixed letter.

A case for type material reference steps emerges from recently reading a section drawn from Migne's Patrology, which has subcolumn numbers. When someone refers to a particular line number, they are inclined to include the subcolumn coordinate along with the line number, but independent of each other, rendering the subcolumn number otiose, e.g., 957c31. There is a less common approach, to restart lineation with each subcolumn, e.g., 957c1, which is a superior method overall. But it might need to be explicitly championed.

A work WF consists only of a sequence of references (a unit or a range), without a text fragment.

Any reference in a scriptum WF and a constrained scriptum WF may be modified further through a [Definition: *text fragment*, applied to a reference unit to constrains the scope to a specific contiguous string]. A text fragment is built primarily [Definition: through *tokens*, with a token being defined as contiguous non-space characters.] ([Definition: *Space characters* are defined as `#x9`, `#x10`, `#x13`, `#x20`.]) The token is then followed by [Definition: an *instance filter*, which specifies by integer which instance of a token is intended,] then perhaps followed by [Definition: a *character filter*, which specifies by integer which characters in the token are intended].

Every text fragment begins with two colons, followed by the token itself. Matches on tokens are case-sensitive, and do not involve any kind of Unicode normalization. If the token includes characters that are reserved (e.g., `$`), they must be escaped.

Note

Unfortunately, the requirement to match based on space-tokenized values poses a problem for constructing WF URIs for writings in languages that use such spaces rarely, if ever (e.g., Chinese). A future version of the WF specifications may provide a better support for languages that do not rely on white space. At present, the entire string must be treated as a single token.

The instance filter, identifying the ordinal position of the token within the reference unit, is marked by square brackets surrounding an integer, e.g., `[1]`, `[36]`.

The character filter is marked by square brackets surrounding an integer, followed perhaps by a hyphen and another integer, e.g., `[1]`, `[3-6]`.

A text fragment may be applied to any level within a reference tree. It does not need to be applied only to leaf references.

Defined Syntax

```
divTypePrefix ::= "n"
ReferenceStep ::= divTypePrefix? INTEGER ( "." INTEGER )?

ReferencesOnly ::= ReferenceOnly ( "&" ReferenceOnly )*
ReferenceOnly ::= ReferenceOnlyUnit | ReferenceOnlyRange
ReferenceOnlyUnit ::= ReferenceStep ( ":" ReferenceStep )*
ReferenceOnlyRange ::= ReferenceOnlyUnit "-" ReferenceOnlyUnit

ReferencesWithText ::= ReferenceWithText ( "&" ReferenceWithText ) *
ReferenceWithText ::= ReferenceWithTextUnit | ReferenceWithTextRange
ReferenceWithTextUnit ::= ReferenceOnlyUnit TextRefAtom?
ReferenceWithTextRange ::= ReferenceWithTextUnit "-" ReferenceWithTextUnit

TextRefAtom ::= "::" TokenVal TokenPosAtom CharPosAtom?
```

```
TokenVal ::= LitFragUnicodeChar+
TokenPosAtom ::= FilterExpItem
CharPosAtom ::= FilterExpItem | FilterExpRange
FilterExpItem ::= "[" INTEGER "]"
FilterExpRange ::= "[" INTEGER "-" INTEGER "]"
```

Conformance

Conformance

The manner in which fragments are retrieved is implementation dependent. Results in the output, however, *must* respect the order in which they appear in the WF.

It is a dynamic error if only one of two references in a range are found. The level of severity and the method of response is implementation-dependent.

In a given WF reference and a given WF-conformant media type item, the depth of steps may not synchronize exactly. If reference steps used to traverse the tree terminates in a media branch node, with remaining subdivisions in the media fragment, the whole fragment must be returned. On the other hand, the media fragment reaches a leaf node (a datum without subdivisions) but there remain reference steps to consume, the empty set *must* be returned for the entire WF URI. A processor *may* provide associated diagnostic information.

Warning

Should there be leniency, when only one simple reference in a complex reference fails?

If, after a match has been found, a filter expression does not match an instance number or a character, the empty set *must* be returned for the entire WF URI.

A designer of WF-conformant media *may* create a mechanism to permit a data creator to make, or to point to, a map or other device that permits one reference system to be converted into another. Whether that mechanism is used is implementation-dependent.

In a constrained scriptum WF, when there is a match on a text fragment, a processor *may* infer that the pointer *might* have relevance for other versions of the same work in the same language, and initiate a process that looks for related constrained scripta. Such an inferential process is the responsibility of the processor, and any resultant data *must* be attributed to the processor. The format of the attribution is processor-dependent, but the attribution must include the URI that uniquely identifies the processor that made the inference.

Every token value in a WF reference is case-sensitive. A WF-processor *must not* case-normalize any WF reference token values.

Strategies for Writing a WF URI

Important

This section is descriptive.

Anyone who writes a WF URI is attempting to point to a portion of text in such a way that it will match the greatest number of WF-conformant files, including those that have not yet been created. Anyone who creates and edits a WF-conformant file intends the content to be matched against the maximum number of corresponding WF URIs. Both parties have it in their best interests to adopt common strategies. Creators of WF-conformant media should note those strategies, to assist match-making.

Every WF URI has one or two URI components that point to a scriptum, and perhaps one URI component that points to a work. Picking an appropriate URI has challenges.

URI vocabularies for scripta are in abundance. As the accompanying examples show, a number of library catalogs furnish unique URIs for scripta such as books, e.g., WorldCat, the Library of Congress, Harvard. There is no shortage of URIs for published scripta. In fact, WorldCat, because it aggregates library data from around the world, frequently offers multiple entries for the same scriptum, and therefore multiple valid URIs. The chief problem is a **wealth of URIs**. The ideal strategies are as follows for the different agents:

- The writer of a WF URI cannot supply alternative URIs for a scriptum. They *must* choose one URI. They *should* adopt the URI they think most common and stable. Avoid URIs that point not to the scriptum but to a digital surrogate, e.g., `ark:` identifiers for scans of the scriptum.
- The creator of WF-conformant data, tethering their data to one or more scriptum URIs, *should* provide as many synonymous instances of a URI as the data format permits, to maximize matches.
- The designer of a WF-conformant media type *should* design the format to facilitate the reconciliation of synonymous URIs.
- The creator of a WF processor *may* introduce methods to attempt creative reconciliation of URI synonyms, as a fallback when synonimity mechanisms in the format fail to produce a match. Whether such methods are used should be controllable by the user.

On the other hand, URI vocabularies for works are meager. Most of the examples of work URIs are drawn from dbpedia, a shadow URI vocabulary based on Wikipedia, which has entries for only the most culturally significant works. Works that are not famous will not have a Wikipedia entry, and therefore lack a dbpedia URI. For modern articles, a DOI as a work URI might be useful, but one can never know if that DOI specifies a work, a scriptum, or both. Many times, a general work URI is available, but what is really needed is a URI not for a work in general but for a specific version. The chief problem here is a **shortage of URIs**. The ideal strategies are as follows:

- The writer of a WF URI should draw from the most common URI vocabulary. They *should not* settle for a general work URI when they really mean a more specific work-version. In many cases a work-version URI must be coined by a user. If the user is intending to coin many work-version URIs and distribute them, they should develop a strategy for easy adoption of the URI vocabulary. If the user is not, then any coined work-version URI will become a niche identifier. To enhance the chances of further interoperability, the following strategy may be adopted for specifying a work-version URI.
  - Embed the work version URI as a tag URN [https://en.wikipedia.org/wiki/Tag_URI_scheme], with a namespace identifying the person or organization coining the URN, e.g., `tag:example.com,2014:`.
  - The very next component should be `$wf0:work:`. This declares that what follows should be taken as designating a work.
  - If the work-version is a version ultimately of another primary work, and that work has a common URI name, let the next portion be that URI, otherwise, build a name for the work from steps that seem appropriate, via a set of `/`-delimited steps that identify the work-version.

  ## Warning

  The advice above has problems, I know. See my earlier comment about work-version URIs.

  The creator of a WF URI *may not* singlehandedly modify someone else's work URI to restrict it to a specific version. However, that work URI *may* be modified if embedded as the component of another URI, as advised above. The tag URN is ideal, because unlike `http:`-style URIs it includes provenance of the writer of the URI. If you are presented with a WF-style tag URN that you wish to modify, replace the namespace component with your own, then modify the portion after `$wf0:work:` as you see fit.
- The creator of WF-conformant data, tethering their data to one or more work URIs, *should* provide as many synonymous instances of a URI as the data format permits, to maximize matches.

- The designer of a WF-conformant media type *should* design the format to permit the reconciliation of URI synonyms.
- The creator of a WF processor *may* introduce methods to attempt reconciliation of URI synonyms. The application of such methods should be controllable by the user.
- Those who mint and publish collections of work-version URIs *should* develop a system that allows a work URI to be modified to restrict the scope to specific versions. Such URIs can be coined only by the owner of the URI scheme. Alternatively, such a service *may* define a syntax that allows third parties to extend the URIs when enjambed in another URI, as suggested above regarding tag URNs.

Warning

See my warning much earlier on work-version URIs.

The writer of a WF URI and the creator of WF-conformant data have a unique shared responsibility to be diligent scriptum readers. Testing a scriptum for WF-viability, and identifying the key reference systems, cannot be performed computationally, and therefore results cannot be computationally validated. Ideally, when declaring a reference system, both parties should point to scripta that are well known. Those reference scripta of course must also yield a clear work-scriptum map (see definitions above). If a scriptum does not yield such a map, it should not be used as a reference scriptum.

Those same two parties have a unique shared responsibility to understand shared conventions for a particular work or scriptum. If shared customs rely primarily upon logical references (e.g., poetry), those should be used; if material references (e.g., modern articles and books), those should be used; if both types are used (e.g., much ancient prose), adopt the system that maximizes precision and familiarity. There is no point in creating a WF URI, or in encoding textual data, through a rarely used reference system.

Writing a WF URI is done most often when building an RDF triple, that is, in making some kind of assertion about the target text. Constructing a strong WF URI for an RDF statement can be enhanced by asking several questions:

1. Do I meant to claim something about a work-version as a whole (as opposed to merely about a particular scriptum)?
2. If yes to #1, do I intend the claim to apply to every version (as opposed to just certain scripta)?
3. Am I making a claim about a particular word or phrase in the text?
4. Am I interested in only a specific work in a particular scriptum?

If #1 is no, then you should build a scriptum WF, with the base URI pointing to a scriptum.

If #1 is yes, but #2 is no, then you should build one constrained scriptum WF per scriptum, serving as the base URI.

If #1, #2, and #3 are all yes, there is a problem. WF URIs do not permit text-based assertions about works in general, because a work is an abstract entity encompassing all versions, including versions in other languages, so there is no single text to quote. The closest you can get is to build a constrained scriptum WF based upon the scriptum that has the best or most popular version of the work in question.

If #1 and #2 are yes, and #3 is no, then you should build a work WF, with the base URI pointing to the work-version.

If #4 is yes, then you are interested in a constrained scriptum WF, which begins with the scriptum as the base URI, then declares what work-version is intended. There must be only one instance of that work-version in the scriptum. If there is more than one, then the work URI needs to be narrowed in scope so that it clearly refers to only that particular version. Further, if you answered yes to #4, then you might have different responses to the previous questions. Keep in mind that some users of your WF URI *may* use it to infer other things about the work-version in other scripta. But strictly speaking,

such inference is not built into the WF URI itself. A constrained scriptum WF points exclusively to a single version of a single work in a single scriptum. Any inferences are the responsibility of the person or algorithm making the inference.

Once you have determined the type of WF you are creating, the construction of the first part of the URI should be clear. The following table shows how the WF URI is started.

Table 2. WF URIs, part 1

WF Type	WF URI
work WF	`<[base URI]#$lf0:a=w;...>`
scriptum WF	`<[base URI]#$lf0:a=s;...>`
constrained scriptum WF	`<[base URI]#$lf0:a=s;w=[work URI];...>`

The next questions pertain to reference systems. The WF will later use numbers that will be used to navigate a reference system. But which reference system are you using?

5. Is the reference system based on physical aspects of a book/article, e.g., pages and lines (as opposed to conceptual units)?

6. Does the base URI point to a book/article that clearly and unambiguously marks the text with this reference system?

7. If not, what book/article unambiguously best states or supports it?

If the answer to #1 is yes, you are using a reference system based on material units; otherwise you are using logical units.

If the answer to #2 is yes, then the reference scriptum may be given the default value (`r=.;`). If no, then #3 should be answered in light of editions of the work that clearly and unambiguously feature the reference system. Ideally, the reference scriptum should be a well-known edition or manuscript that has only one version of the work, and clearly has only one material reference system or one logical reference system (or only one of each). If the URI you use for the reference scriptum has the # in it, be sure to convert it to `%23`.

Table 3. WF URIs, part 2

Reference System	Best Reference Scriptum	WF URI
material units	Base URI	`<...t=m;r=.;...>`
logical units	Base URI	`<...t=l;r=.;...>`
material units	Another scriptum	`<...t=m;r=[scriptum URI];...>`
logical units	Another scriptum	`<...t=l;r=[scriptum URI];...>`

In many cases the choice of the reference system should be relatively easy. In general, use the number-based reference system you would use if you were citing the item in ordinary prose. Custom frequently rests upon the type of work or scriptum that is being identified. Below are some common cases, along with ideal strategies.

Warning

The advice below is starting to overlap with similar advice that appears in the examples [./wf-examples.xml], where most of my energy is going now, in the hopes that they will become rich enough to supplant, or at least enrich, the following strategies. That XML file is much easier to develop and maintain than a docbook file.

- **Printed article, book, or book section**. If the item is clearly paginated, use the material reference system based upon page and line number or page and note number. If the item consistently has two or more columns, be sure to use column numeration. Note, some books (e.g., Migne's Patrology) invest columns with the topmost numeration system, not pages, and sometimes enumerate subcolumn positions. Let the labels in the paratext be your primary guide.

  Similarly, be aware of what scriptum your reference URI points to. If it points to a multivolume scriptum, then the reference system *must* begin with volume numbers. But if it points to a specific volume in the multivolume work, then the reference system *must* begin with the page numeration system.

  Similarly, if the article is part of a newspaper that is first organized by section then by page, the reference system should follow that structure. In many cases, a newspaper will label and order sections through the alphabet, sometimes with significant gaps because the publishers wish to pick meaningful letters. For example, a newspaper's sections might run A, B, C, S (= style), Z (= classified advertisements). In these cases the sequence is ordered but selective, and the labels should be converted to integers that reflect the sequence in the newspaper, not the alphabet (i.e., 1, 2, 3, 4, 5 not 1, 2, 3, 19, 26).

  Any clearly labeled logical reference system, may be used for an item, if there are no other competing logical reference systems evident.

- **Article, post, or comment published in HTML**. Most of these scripta are not candidates for WF URIs. The item must be uniquely named, and have explicitly labeled sections or paragraphs. There is no support for a material reference system in HTML or any other digital format. Absent an explicitly labeled logical reference system, the item is not a candidate for WF URIs. Consider using XPointer instead.

  Possible exception: An archival service may create a copy of the HTML page and superimpose upon it an explicit labeling system. In these cases, the item is a candidate for WF URIs.

  Possible exception: Someone may develop a set of constraints to create a WF-conformant HTML media format. There is no such known media format, and none are planned. If it is created then items in that constrained HTML format would be candidates for WF URIs.

- **Email, text messages, and other digital media**. See previous point, regarding HTML resources. An exception may be made for PDFs or other formats that can be relied upon to accurately represent format and presentation. Such media should be treated as if written on material media. References will be made not to the underlying encoding of the file, but to its visible features.

For other strategies, see the next section, and annotations attached to specific examples.

Examples

Important

This section is descriptive.

See wf-examples.xml [./wf-examples.xml] for an XML file that presents several examples of WF URIs, with commentary. At some point in the future, those examples will be better integrated with these specifications, and perhaps a quick-start tutorial.

Scrap

Warning

This is material I wrote, then thought not worth keeping. But I wasn't quite ready to delete it, either. So it's here, perhaps only for the record.

Defining or declaring the relationship between one scriptum/work and another is also outside the scope of WF URIs. Any WF-conformant data format, or processor of WF-conformant data, *may* define mechanisms for resolving and inferring relationships between scripta and works. Anyone who builds a WF URI should be aware that users of that URI may infer correspondences or relationships not explicitly stated. Such inferences are made all the time in the Semantic Web. Some of them are justifiable, others not. The WF guidelines are written with the understanding that inferences are inevitable, and are the responsibility of those making the inferences. It is recommended that any RDF triple involving WF URIs be credited to the person(s) or algorithm(s) making the assertion. That is, it is recommended that triples be reified (treated as an object in its own right) and assigned provenance. The conventions of RDF* [https://github.com/w3c/rdf-star] are recommended to represent triples involving WF URIs.

In the RDF Turtle format, any kind of URI string may be assigned a prefix, and the task of manually writing WF URIs may be simplified as follows:

```
@prefix db: <http://dbpedia.org/resource/> .
@prefix cito: <http://purl.org/spar/cito/> .
@prefix iliad: <http://dbpedia.org/resource/Iliad#$lf0:a=w;r=http://www.worldca

db:Republic_(Plato) cito:cites iliad:1.1$
```

Warning

I'm not certain if Turtle allows @prefix to extend beyond the base URI into the fragment part, as I've suggested above. If so, that's great, because it would avoid a lot of repeated.