

---

# **MyCapytains Documentation**

*Release 0.0.1*

**Thibault Clérice**

**Jan 30, 2020**



---

# Contents

---

|          |   |            |
|----------|---|------------|
| <b>1</b> | <b>Simple Example of what it does</b>           | <b>3</b>   |
| <b>2</b> | <b>Installation and Requirements</b>            | <b>5</b>   |
| <b>3</b> | <b>Contents</b>                                 | <b>7</b>   |
| 3.1      | MyCapytain's Main Objects Explained . . . . .   | 7          |
| 3.2      | Project using MyCapytain . . . . .              | 20         |
| 3.3      | Working with Local CapiTainS XML File . . . . . | 21         |
| 3.4      | Known issues and recommendations . . . . .      | 22         |
| 3.5      | MyCapytain API Documentation . . . . .          | 22         |
| 3.6      | Benchmarks . . . . .                            | 126        |
|          | <b>Python Module Index</b>                      | <b>127</b> |
|          | <b>Index</b>                                    | <b>129</b> |



MyCapytain is a python library which provides a large set of methods to interact with Text Services API such as the Canonical Text Services, the Distributed Text Services. It also provides a programming interface to exploit local textual resources developed according to the Capitains Guidelines.



---

## Simple Example of what it does

---

The following code and example is badly displayed at the moment on Github. We recommend you to go to <http://mycapytain.readthedocs.org>

On Leipzig DH Chair's Canonical Text Services API, we can find the Epigrammata of Martial. This texts are identified by the identifier "urn:cts:latinLit:phi1294.phi002.perseus-lat2". We want to have some information about this text so we are gonna ask the API to give its metadata to us :

Listing 1: example.py from the Github Repository

```

1  from MyCapytain.resolvers.cts.api import HttpCtsResolver
2  from MyCapytain.retrievers.cts5 import CTS
3  from MyCapytain.common.constants import Mimetypes
4
5  # We set up a resolver which communicates with an API available in Leipzig
6  resolver = HttpCtsResolver(CTS("http://cts.dh.uni-leipzig.de/api/cts/"))
7  # We require some metadata information
8  textMetadata = resolver.getMetadata("urn:cts:latinLit:phi1294.phi002.perseus-lat2")
9  # Texts in CTS Metadata have one interesting property : its citation scheme.
10 # XmlCtsCitation are embedded objects that carries information about how a text can
    ↳ be quoted, what depth it has
11 print(type(textMetadata), [citation.name for citation in textMetadata.citation])

```

This query will return the following information :

```
<class 'MyCapytain.resources.collections.cts.Text'> ['book', 'poem', 'line']
```

```

12 # Now, we want to retrieve the first line of poem seventy two of the second book
13 passage = resolver.getTextualNode("urn:cts:latinLit:phi1294.phi002.perseus-lat2",
    ↳ subreference="2.72.1")
14 # And we want to have its content exported to plain text and have the siblings of
    ↳ this passage (previous and next line)
15 print(passage.export(Mimetypes.PLAINTEXT), passage.siblingsId)

```

And we will get

```
Hesterna factum narratur, Postume, cena
```

If you want to play more with this, like having a list of what can be found in book three, you could go and do

```
16 poemsInBook3 = resolver.getReffs("urn:cts:latinLit:phi1294.phi002.perseus-lat2",  
17 ↪subreference="3")  
17 print (poemsInBook3)
```

Which would be equal to :

```
['3.1', '3.2', '3.3', '3.4', '3.5', '3.6', '3.7', '3.8', '3.9', '3.10', '3.11', '3.12'  
↪, '3.13', ...]
```

Now, it's your time to work with the resource ! See the CapiTainS Classes page on ReadTheDocs to have a general introduction to MyCapytain objects !

---

### Installation and Requirements

---

The best way to install MyCapytain is to use pip. MyCapytain tries to support Python over 3.4.

The work needed for supporting Python 2.7 is mostly done, however, since 2.0.0, we are giving up on ensuring that MyCapytain will be compatible with Python < 3 while accepting PR which would help doing so.

```
pip install MyCapytain
```

If you prefer to use setup.py, you should clone and use the following

```
git clone https://github.com/Capitains/MyCapytain.git
cd MyCapytain
python setup.py install
```



## 3.1 MyCapytain's Main Objects Explained

### 3.1.1 Exportable Parent Classes

#### Description

`MyCapytain.common.base.Exportable`

The `Exportable` class is visible all across the library. It provides a common, standardized way to retrieve in an API fashion to what can an object be exported and to exports it. Any exportable object should have an `EXPORT_TO` constant variable and include a `__export__(output, **kwargs)` methods if it provides an export type.

#### Example

The following code block is a mere example of how to implement `Exportable` and what are its responsibilities. Exportable typically loops over all the parents class of the current class until it find one exportable system matching the required one.

```
1 from MyCapytain.common.constants import Mimetypees
2 from MyCapytain.common.base import Exportable
3
4
5 class Sentence(Exportable):
6     """ This class represent a Sentence
7
8     :param content: Content of the sentence
9     """
10    # EXPORT_TO is a list of Mimetype the object is capable to export to
11    EXPORT_TO = [
12        Mimetypees.PLAINTEXT, Mimetypees.XML.Std
13    ]
```

(continues on next page)

(continued from previous page)

```

14 DEFAULT_EXPORT = Mimetypes.PLAINTEXT
15
16 def __init__(self, content):
17     self.content = content
18
19 def __export__(self, output=None, **kwargs):
20     """ Export the collection item in the Mimetype required.
21
22     :param output: Mimetype to export to (Uses MyCapytain.common.utils.Mimetypes)
23     :type output: str
24     :return: Object using a different representation
25     """
26     if output == Mimetypes.PLAINTEXT:
27         return self.content
28     elif output == Mimetypes.XML.Std:
29         return "<sentence>{}</sentence>".format(self.content)
30
31
32 class TEISentence(Sentence):
33     """ This class represent a Sentence but adds some exportable accepted output
34
35     :param content: Content of the sentence
36     """
37     EXPORT_TO = [
38         Mimetypes.JSON.Std
39     ]
40
41 def __export__(self, output=None, **kwargs):
42     """ Export the collection item in the Mimetype required.
43
44     :param output: Mimetype to export to (Uses MyCapytain.common.utils.Mimetypes)
45     :type output: str
46     :return: Object using a different representation
47     """
48     if output == Mimetypes.JSON.Std:
49         return {"http://www.tei-c.org/ns/1.0/sentence": self.content}
50     elif output == Mimetypes.XML.Std:
51         return "<sentence xmlns=\"http://www.tei-c.org/ns/1.0\">{}</sentence>".
52         ↪format(self.content)
53
54 s = Sentence("I love Martial's Epigrammatas")
55 print(s.export(Mimetypes.PLAINTEXT))
56 # I love Martial's Epigrammatas
57 print(s.export()) # Defaults to PLAINTEXT
58 # I love Martial's Epigrammatas
59 print(s.export(Mimetypes.XML.Std))
60 # <sentence>I love Martial's Epigrammatas</sentence>
61
62 tei = TEISentence("I love Martial's Epigrammatas")
63 print(tei.export(Mimetypes.PLAINTEXT))
64 # I love Martial's Epigrammatas
65 print(tei.export()) # Defaults to PLAINTEXT
66 # I love Martial's Epigrammatas
67 print(tei.export(Mimetypes.JSON.Std))
68 # {"http://www.tei-c.org/ns/1.0/sentence": I love Martial's Epigrammatas}
69 print(tei.export(Mimetypes.XML.Std)) # Has been rewritten by TEISentence

```

(continues on next page)

(continued from previous page)

```

70 # <sentence xmlns="http://www.tei-c.org/ns/1.0">I love Martial's Epigrammatas</
    ↳sentence>
71 try:
72     print (tei.export (Mimetypes.XML.RDF) )
73 except NotImplementedError as error:
74     print (error)
75 # Raise the error and prints "Mimetype application/rdf+xml has not been implemented,
    ↳for this resource"

```

### 3.1.2 Retrievers

MyCapytain.retrieveers.prototypes.API

#### Description

Retrievers are classes that help build requests to API and return standardized responses from them. There is no real perfect prototypes. The only requirements for a Retriever is that its query function should returns string only. It is not the role of the retrievers to parse response. It is merely to facilitate the communication to remote API most of the time.

#### Recommendations

For Textual API, it is recommended to implement the following requests

- getTextualNode(textId[str], subreference[str], prevnext[bool], metadata[bool])
- getMetadata(objectId[str], \*\*kwargs)
- getSiblings(textId[str], subreference[str])
- getReffs(textId[str], subreference[str], depth[int])

#### Example of implementation : CTS 5

MyCapytain.retrieveers.cts5.HttpCtsRetriever

```

1  from MyCapytain.retrieveers.cts5 import HttpCtsRetriever
2
3  # We set up a retriever which communicates with an API available in Leipzig
4  retriever = HttpCtsRetriever("http://cts.dh.uni-leipzig.de/api/cts")
5  # We require a passage : passage is now a Passage object
6  passage = retriever.getPassage("urn:cts:latinLit:phil294.phil002.perseus-lat2:1.1")
7  # Passage is now equal to the string content of http://cts.dh.uni-leipzig.de/api/cts/?
    ↳request=GetPassage&urn=urn:cts:latinLit:phil294.phil002.perseus-lat2:1.1
8  print (passage)
9
10  """
11  <GetPassage><request><requestName>GetPassage</requestName><requestUrn>
    ↳urn:cts:latinLit:phil294.phil002.perseus-lat2:1.1</requestUrn></request>
12  <reply><urn>urn:cts:latinLit:phil294.phil002.perseus-lat2:1.1</urn><passage><TEI>
13  <text n="urn:cts:latinLit:phil294.phil002.perseus-lat2" xml:id="stoa0045.stoa0"><body>
14  <div type="edition" n="urn:cts:latinLit:phil294.phil002.perseus-lat2" xml:lang="lat">
15  <div type="textpart" subtype="book" n="1"><div type="textpart" subtype="poem" n="1">
16  <head>I</head>

```

(continues on next page)

(continued from previous page)

```

17 <l n="1">Hic est quem legis ille, quem requiris, </l>
18 <l n="2">Toto notus in orbe Martialis </l>
19 <l n="3">Argutis epigrammaton libellis: <pb/></l>
20 <l n="4">Cui, lector studiose, quod dedisti </l>
21 <l n="5">Viventi decus atque sentienti, </l>
22 <l n="6">Rari post cineres habent poetae. </l>
23 </div></div></div></body></text></TEI></passage></reply>
24 ""

```

### 3.1.3 Text and Passages

#### Description

#### Hierarchy

The generic idea of both Text and Passage's classes is that they inherit from a longer trail of text bearing object that complexified over different features. The basic is

- *TextualElement* is an object which can bear Metadata and Collection information. It has a `.text` property and is exportable
- *TextualNode* inherits from `NodeId` and unlike *TextualElement*, *TextualNode* is part of a graph of `CitableObject`. It bears informations about its siblings, parents, children.
- *TextualGraph* is a bit interactive : you can query for children nodes and get descendant references of the object.
- *InteractiveTextualNode* is completely interactive . You can browse the graph by accessing the `.next` property for example : it should then return an *InteractiveTextualNode* as well
- *CtsNode* has two unique methods more as well as a `urn` property.
- From *CtsNode* we find *CitableText* and *Passage*, which represents complete and portion of a Text. The main difference is that *CitableText* has no parents, no siblings.

Fig. 1: Prototype of Texts from `:module:'MyCapytain.resources.prototypes.text'`. `NodeId` and `Exportable` are respectively from `:module:'MyCapytain.common.reference'` and `:module:'MyCapytain.common.base'`.

#### Objectives

Text and Passages object have been built around *InteractiveTextualNode* which fills the main purpose of MyCapytain : being able to interact with citable, in-graph texts that are retrieve through web API or local files. Any implementation should make sure that the whole set of navigation tool are covered. Those are :

| Tree Identifiers(Returns str Identifiers) | Tree Navigations (Returns InteractiveTextualNode or children class) | Retrieval Methods                | Other   |
|---|---|----------------------------------|---|
| prevId                                    | prev  | .getTextualNode(subreference)    | id : TextualNode Identifier [str]                           |
| nextId                                    | nextId  | .getRefs(subreference[optional]) | metadata : Metadata information [Metadata]                  |
| siblingsId [tuple[str]]                   | siblings [tuple[InteractiveTextualNode]]                            |                                  |   |
| parentId                                  | parent  |                                  | citation : Citation Information [Citation]                  |
| childIds [list[str]]                      | children [list[InteractiveTextualNode]]                             |                                  | text : String Representation of the text without annotation |
| firstId                                   | first   |                                  | .export()   |
| lastId                                    | last  |                                  |   |

## The base module

The base module contains special implementations : they technically do not support interactive methods but provides generic parsing and export methods for specific type of contents such as TEI XML object or other formats such as json, csv, treebank objects in the future.

The TEIResource for example requires the object to be set up with a resource parameters that will be furtherparsed using lxml. From there, it provides export such as plain/text, TEI XML, nested dictionaries or even anlxml etree interface.

## Implementation example : HTTP API Passage work

```

1 from MyCapytain.retrieveers.cts5 import HttpCtsRetriever
2 from MyCapytain.resources.texts.remote.cts import CtsText
3
4 # We set up a retriever which communicates with an API available in Leipzig
5 retriever = HttpCtsRetriever("http://cts.dh.uni-leipzig.de/api/cts")
6
7 # Given that we have other examples that shows how to work with text,
8 # we will focus here on playing with the graph functionality of texts implementations.
9 # We are gonna retrieve a text passage and the retrieve all its siblings in different
10 ↪fashion#
11 # The main point is to find all children of the same parent.
12 # The use case could be the following : some one want to retrieve the full text
13 ↪around a citation
14 # To enhance the display a little.
15
16 # We will work with the line 7 of poem 39 of book 4 of Martial's Epigrammata
17 # The text is urn:cts:latinLit:phi1294.phi002.perseus-lat2
18 text = CtsText(retriever=retriever, urn="urn:cts:latinLit:phi1294.phi002.perseus-lat2
19 ↪")
20
21 # We retrieve up the passage
22 target = text.getTextualNode(subreference="4.39.7")
23 print(target.text)
24 """
25 Nec quae Callaico linuntur auro,
```

(continues on next page)

```

23 """
24
25 # The parent way :
26 # - get to the parent,
27 # - retrieve each node,
28 # - print only the one which are not target
29
30 parent = target.parent
31 for node in parent.children:
32     if node.id != target.id:
33         print("{}\t{}".format(node.id, node.text))
34     else:
35         print("-----Original Node-----")
36
37 """
38 4.39.1     Argenti genus omne comparasti,
39 4.39.2     Et solus veteres Myronos artes,
40 4.39.3     Solus Praxitelus manum Scopaeque,
41 4.39.4     Solus Phidiaci toreuma caeli,
42 4.39.5     Solus Mentoreos habes labores.
43 4.39.6     Nec desunt tibi vera Gratiana,
44 -----Original Node-----
45 4.39.8     Nec mensis anaglypta de paternis.
46 4.39.9     Argentum tamen inter omne miror
47 4.39.10    Quare non habeas, Charine, purum.
48 """
49
50 print("\n\nSecond Method\n\n")
51
52 # We are gonna do another way this time :
53 # - get the previous until we change parent
54 # - get the next until we change parent
55
56 parentId = node.parentId
57 # Deal with the previous ones
58 current = target.prev
59 while current.parentId == parentId:
60     print("{}\t{}".format(current.id, current.text))
61     current = current.prev
62
63 print("-----Original Node-----")
64
65 # Deal with the next ones
66 current = target.next
67 while current.parentId == parentId:
68     print("{}\t{}".format(current.id, current.text))
69     current = current.next
70
71 """
72 4.39.6     Nec desunt tibi vera Gratiana,
73 4.39.5     Solus Mentoreos habes labores.
74 4.39.4     Solus Phidiaci toreuma caeli,
75 4.39.3     Solus Praxitelus manum Scopaeque,
76 4.39.2     Et solus veteres Myronos artes,
77 4.39.1     Argenti genus omne comparasti,
78 -----Original Node-----
79 4.39.8     Nec mensis anaglypta de paternis.
80 4.39.9     Argentum tamen inter omne miror

```

(continues on next page)

(continued from previous page)

```

80 4.39.10      Quare non habeas, Charine, purum.
81
82 """

```

### Other Example

See MyCapytain.resources.texts.local

## 3.1.4 Collection

### Description

Collections are the metadata containers object in MyCapytain. Unlike other object, they will never contain textual content such as Texts and Passages but will in return help you browse through the catalog of one APIs collection and identify manually or automatically texts that are of relevant interests to you.

The main information that you should be interested in are :

- Collections are children from Exportable. As of 2.0.0, any collection can be exported to JSON DTS.
- Collections are built on a hierarchy. They have children and descendants
- Collections have identifiers and title (Main name of what the collection represents : if it's an author, it's her name, a title for a book, a volume label for a specific edition, etc.)
- Collections can inform the machine if it represents a readable object : if it is readable, it means that using its identifier, you can query for passages or references on the same API.

### Main Properties

- Collection().id : Identifier of the object
- Collection().get\_label(lang[Optional]) : Title of the object
- Collection().readable : If True, means that the Collection().id can be used in GetReffs or GetTextualNode queries
- Collection().members : Direct children of the object
- Collection().descendants : Direct and Indirect children of the objects
- Collection().readableDescendants : Descendants that have .readable as True
- Collection().export() : Export Method
- Collection().metadata : Metadata object that contain flat descriptive localized information about the object.

### Implementation : CTS Collections

---

**Note:** For a recap on what Textgroup means or any CTS jargon, go to <http://capitains.github.io/pages/vocabulary>

---

CTS Collections are divided in 4 kinds : TextInventory, TextGroup, Work, Text. Their specificity is that the hierarchy of these objects are predefined and always follow the same order. They implement a special export (MyCapytain.common.constants.Mimetypes.XML.CTS) which basically exports to the XML Text Inventory Format that one would find making a GetCapabilities request.

CapiTainS CTS Collections implement a parents property which represent a list of parents where .parents' order is equal to `Text.parents = [Work(), TextGroup(), TextInventory()]`.

Their finale implementation accepts to parse resources through the `resource=` named argument.

### Example

```
1 from MyCapytain.retrievers.cts5 import HttpCtsRetriever
2 from MyCapytain.resources.collections.cts import XmlCtsTextInventoryMetadata, \
  ↪ XmlCtsWorkMetadata
3 from MyCapytain.common.constants import Mimetypes
4 from pprint import pprint
5
6 """
7 In order to have a real life example,
8 we are gonna query for data in the Leipzig CTS API
9 We are gonna query for metadata about Seneca who
10 is represented by urn:cts:latinLit:stoa0255
11
12 To retrieve data, we are gonna make a GetMetadata query
13 to the CTS Retriever.
14 """
15 retriever = HttpCtsRetriever("http://cts.dh.uni-leipzig.de/api/cts")
16 # We store the response (Pure XML String)
17 response = retriever.getMetadata(objectId="urn:cts:latinLit:stoa0255")
18
19 """
20 From here, we actually have the necessary data, we can now
21 play with collections. TextInventory is the main collection type that is needed to
22 parse the whole response.
23 """
24 inventory = XmlCtsTextInventoryMetadata.parse(resource=response)
25 # What we are gonna do is print the title of each descendant :
26 for descendant in inventory.descendants:
27     # Metadatum resolve any non-existing language ("eng", "lat") to a default one
28     # Putting default is just making that clear
29     print(descendant.get_label())
30
31 """
32 You should see in there things such as
33 - "Seneca, Lucius Annaeus" (The TextGroup or main object)
34 - "de Ira" (The Work object)
35 - "de Ira, Moral essays Vol 2" (The Edition specific Title)
36
37 We can now see other functions, such as the export to JSON DTS.
38 Collections have a unique feature built in : they allow for
39 accessing an item using its key as if it were a dictionary :
40 The identifier of a De Ira is urn:cts:latinLit:stoa0255.stoa0110
41 """
42 deIra = inventory["urn:cts:latinLit:stoa0255.stoa010"]
43 assert isinstance(deIra, XmlCtsWorkMetadata)
44 pprint(deIra.export(output=Mimetypes.JSON.DTS.Std))
45 # you should see a DTS representation of the work
46
47 """
```

(continues on next page)

(continued from previous page)

```

48 What we might want to do is to browse metadata about seneca's De Ira
49 Remember that CtsCollections have a parents attribute !
50 """
51 for descAsc in deIra.descendants + [deIra] + deIra.parents:
52     # We filter out Textgroup which has an empty Metadata value
53     if not isinstance(descAsc, HttpCtsRetriever):
54         print (
55             descAsc.metadata.export (output=Mimetypes.JSON.Std)
56         )
57 """
58 And of course, we can simply export deIra to CTS XML format
59 """
60 print (deIra.export (Mimetypes.XML.CTS))

```

### 3.1.5 Dispatchers

#### Description

Dispatcher are tools to be used to organize semi-automatically resources, most likely in the context of a resolver retrieving data from local directories.

Dispatcher has following properties :

- `methods` -> [tuple(Callable()->Bool, str)]
- `add(func[Callable()], inventory_name[str])`
- `inventory(inventory_name[str])` -> @decorator
- `dispatch(collection[Collection, **kwargs])`

#### Implementation

```
class MyCapytain.resolvers.utils.CollectionDispatcher (collection, de-
fault_inventory_name=None)
```

Collection Dispatcher provides a utility to divide automatically texts and collections into different collections

##### Parameters

- **collection** – The root collection
- **default\_inventory\_name** – The default name of the default collection

```
add (func, inventory_name)
```

Register given function as a filter.

If this function “func” returns True when given an object, said object will be dispatched to `Collection(inventory_name)`

##### Parameters

- **func** – Callable
- **inventory\_name** – Identifier of the collection to dispatch to

```
dispatch (collection, **kwargs)
```

Dispatch a collection using internal filters

##### Parameters

- **collection** – Collection object
- **kwargs** – Additional keyword arguments that could be used by internal filters

**Returns** None

**Raises**

**inventory** (*inventory\_name*)

Decorator to register filters for given inventory. For a function “abc”, it has the same effect

**Parameters** *inventory\_name* –

**Returns**

```
tic = CtsTextInventoryCollection()
latin = CtsTextInventoryMetadata("urn:perseus:latinLit", parent=tic)
latin.set_label("Classical Latin", "eng")
dispatcher = CollectionDispatcher(tic)

@dispatcher.inventory("urn:perseus:latinLit")
def dispatchLatinLit(collection, path=None, **kwargs):
    if collection.id.startswith("urn:cts:latinLit:"):
        return True
    return False
```

**methods**

List of methods to dispatch resources.

**Each element is a tuple with two elements :**

- First one is the inventory identifier to dispatch to
- Second one is a function which, if returns true, will activate dispatching to given

**Return type** List

## Example

```
1 from MyCapytain.resolvers.cts.local import CtsCapitainsLocalResolver
2 from MyCapytain.resolvers.utils import CollectionDispatcher
3 from MyCapytain.common.constants import Mimetypes
4 from MyCapytain.resources.collections.cts import XmlCtsTextInventoryMetadata
5 from MyCapytain.resources.prototypes.cts.inventory import CtsTextInventoryCollection
6
7 # We set up a main collection
8 tic = CtsTextInventoryCollection()
9 # We register sub collection we want to dispatch to
10 latin = XmlCtsTextInventoryMetadata("urn:perseus:latinLit", parent=tic)
11 latin.set_label("Classical Latin", "eng")
12 farsi = XmlCtsTextInventoryMetadata("urn:perseus:farsiLit", parent=tic)
13 farsi.set_label("Farsi", "eng")
14 gc = XmlCtsTextInventoryMetadata("urn:perseus:greekLit", parent=tic)
15 gc.set_label("Ancient Greek", "eng")
16 gc.set_label("Grec Ancien", "fre")
17
18 # We create the dispatcher with the root collection
19 dispatcher = CollectionDispatcher(tic)
20
```

(continues on next page)

(continued from previous page)

```

21 # And we record function for each given repository
22 # We could have two function dispatching for the same repository !
23 @dispatcher.inventory("urn:perseus:latinLit")
24 def dispatchLatinLit(collection, path=None, **kwargs):
25     if collection.id.startswith("urn:cts:latinLit:"):
26         return True
27     return False
28
29 @dispatcher.inventory("urn:perseus:farsiLit")
30 def dispatchFarsiLit(collection, path=None, **kwargs):
31     if collection.id.startswith("urn:cts:farsiLit:"):
32         return True
33     return False
34
35 @dispatcher.inventory("urn:perseus:greekLit")
36 def dispatchGreekLit(collection, path=None, **kwargs):
37     if collection.id.startswith("urn:cts:greekLit:"):
38         return True
39     return False
40
41 # We set up a resolver which parses local file
42 NautilusDummy = CtsCapitainsLocalResolver(
43     resource=[
44         "./tests/testing_data/latinLit2"
45     ],
46     # We give it the dispatcher
47     dispatcher=dispatcher
48 )
49
50 # If we want to read the main repository, we will have all children
51 all = NautilusDummy.getMetadata()
52
53 print(len(all.readableDescendants)) # 25 is the number of edition and translation
54 print([m.id for m in all.members]) # Direct members are dispatched-in collections
55 print(
56     all["urn:cts:latinLit:phil1294"] == all["urn:perseus:latinLit"][
57     ↪ "urn:cts:latinLit:phil1294"]
58 ) # Is true because they are dispatched this way
59
60 try:
61     all["urn:perseus:greekLit"]["urn:cts:latinLit:phil1294"]
62 except KeyError:
63     print("But this won't work because the object has been dispatched to latinLit !")
64
65 print(len(all["urn:perseus:greekLit"].readableDescendants)) # Is 6 because there is_
66 ↪ 6 recorded texts in __cts__
67 print(len(all["urn:perseus:latinLit"].readableDescendants)) # Is 19 because there is_
68 ↪ 6 recorded texts in __cts__

```

### 3.1.6 Resolvers

#### Description

Resolvers were introduced in 2.0.0b0 and came as a solution to build tools around Text Services APIs where you can seamlessly switch a resolver for another and not changing your code, join together multiple resolvers, etc. The

principle behind resolver is to provide native python object based on API-Like methods which are restricted to four simple commands :

- `getTextualNode(textId[str], subreference[str], prevnext[bool], metadata[bool]) -> Passage`
- `getMetadata(objectId[str], **kwargs) -> Collection`
- `getSiblings(textId[str], subreference[str]) -> tuple([str, str])`
- `getReffs(textId[str], subreference[str], depth[int]) -> list([str])`

These function will always return objects derived from the major classes, *i.e.* Passage and Collection for the two firsts and simple collections of strings for the two others. Resolvers fills the hole between these base objects and the original retriever objects that were designed to return plain strings from remote or local APIs.

The base functions are represented in the prototype, and only `getMetadata` might be expanded in terms of arguments depending on what filtering can be offered. Though, any additional filter has not necessarily effects with other resolvers.

### Historical Perspective

The original incentive to build resolvers was the situation with retrievers, in the context of the Nautilus API and Nemo UI : Nemo took a retriever as object, which means that, based on the prototype, Nemo was retrieving string objects. That made sense as long as Nemo was running with HTTP remote API because it was actually receiving string objects which were not even (pre-)processed by the Retriever object. But since Nautilus was developed (a fully native python CTS API), we had the situation where Nemo was parsing strings that were exported from python etree objects by Nautilus which parsed strings.

Introducing Resolvers, we managed to avoid this double parsing effect in any situation : MyCapytain now provides a default class to provide access to querying text no matter what kind of transactions there is behind the Python object. At the same time, Resolvers provide a now unified system to retrieve texts independently from the retrieverstandard type (CTS, DTS, Proprietary, etc.).

### Prototype

**class** `MyCapytain.resolvers.prototypes.Resolver`

Resolver provide a native python API which returns python objects.

Initiation of resolvers are dependent on the implementation of the prototype

**getMetadata** (*objectId*: *str* = *None*, *\*\*filters*) → `MyCapytain.resources.prototypes.metadata.Collection`  
Request metadata about a text or a collection

#### Parameters

- **objectId** (*str*) – Object Identifier to filter on
- **filters** (*dict*) – Kwarg parameters.

#### Returns

**getReffs** (*textId*: *str*, *level*: *int* = *1*, *subreference*: `Union[str, MyCapytain.common.reference._base.BaseReference]` = *None*, *include\_descendants*: *bool* = *False*, *additional\_parameters*: `Optional[Dict[str, Any]]` = *None*) → `MyCapytain.common.reference._base.BaseReferenceSet`  
Retrieve the siblings of a textual node

**Parameters**

- **textId** (*str*) – CtsTextMetadata Identifier
- **level** (*int*) – Depth for retrieval
- **subreference** (*str*) – CapitainsCtsPassage Reference
- **include\_descendants** –
- **additional\_parameters** –

**Returns** List of references

**Return type** [str]

..todo :: This starts to be a bloated function. ...

```
getSiblings (textId: str, subreference: Union[str, MyCapy-
            tain.common.reference._base.BaseReference]) → Tu-
            ple[MyCapytain.common.reference._base.BaseReference, MyCapy-
            tain.common.reference._base.BaseReference]
```

Retrieve the siblings of a textual node

**Parameters**

- **textId** (*str*) – CtsTextMetadata Identifier
- **subreference** (*str*) – CapitainsCtsPassage Reference

**Returns** Tuple of references

**Return type** (str, str)

```
getTextualNode (textId: str, subreference: Union[str, MyCapy-
                tain.common.reference._base.BaseReference] = None, prevnext: bool = False,
                metadata: bool = False) → MyCapytain.resources.prototypes.text.TextualNode
```

Retrieve a text node from the API

**Parameters**

- **textId** (*str*) – CtsTextMetadata Identifier
- **subreference** (*str*) – CapitainsCtsPassage Reference
- **prevnext** (*boolean*) – Retrieve graph representing previous and next passage
- **metadata** (*boolean*) – Retrieve metadata about the passage and the text

**Returns** CapitainsCtsPassage

**Return type** *CapitainsCtsPassage*

**Example**

```
1 from MyCapytain.resolvers.cts.api import HttpCtsResolver
2 from MyCapytain.retrievers.cts5 import HttpCtsRetriever
3 from MyCapytain.common.constants import Mimetypes, XPATH_NAMESPACES
4
5 # We set up a resolver which communicates with an API available in Leipzig
6 resolver = HttpCtsResolver(HttpCtsRetriever("http://cts.dh.uni-leipzig.de/api/cts"))
7 # We require a passage : passage is now a Passage object
8 # This is an entry from the Smith Myth Dictionary
9 # The inner methods will resolve to the URI http://cts.dh.uni-leipzig.de/api/cts?
  ↪ request=GetPassage&urn=urn:cts:pdlrefwk:viaf88890045.003.perseus-eng1:A.abaeus_1
```

(continues on next page)

(continued from previous page)

```

10 # And parse it into interactive objects
11 passage = resolver.getTextualNode("urn:cts:pdlrefwk:viaf88890045.003.perseus-eng1",
   ↪ "A.abaeus_1")
12 # We need an export as plaintext
13 print(passage.export(
14     output=Mimetypes.PLAINTEXT
15 ))
16 """
17     Abaeus ( βαιο ), a surname of Apollo
18     derived from the town of Abae in Phocis, where the god had a rich temple.
   ↪ (Hesych. s. v.
19     βαιο ; Hdt. 8.33 ; Paus. 10.35.1 , &c.) [ L.S ]
20 """
21 # We want to find bibliographic information in the passage of this dictionary
22 # We need an export as LXML ETREE object to perform XPath
23 print(
24     passage.export(
25         output=Mimetypes.PYTHON.ETREE
26     ).xpath("./tei:bibl/text()", namespaces=XPATH_NAMESPACES, magic_string=False)
27 )
28 ["Hdt. 8.33", "Paus. 10.35.1"]

```

## 3.2 Project using MyCapytain

If you are using MyCapytain and wish to appear here, please feel free to open an [issue](#)

### 3.2.1 Extensions

#### Nautilus

Nautilus provides a local retriever to build inventory based on a set of folders available locally.

#### Flask Capitains Nemo

Flask Capitains Nemo is an extension for Flask to build a browsing interface using both retrievers and resources modules. You will find example of use in a web based environment.

#### HookTest

HookTest is a library and command line tools for checking resources against the Capitains Guidelines You'll find uses mainly in `units.py`

#### CLTK Corpora Converter

Capitains Corpora Converter Converts CapiTainS-based Repository (<http://capitains.github.io>) to JSON for CLTK

## 3.3 Working with Local CapiTainS XML File

### 3.3.1 Introduction

The class `MyCapytain.resources.texts.locals.tei.Text` requires the guidelines of Capitains to be implemented in your file.

### 3.3.2 Example

```

1  # We import the correct classes from the local module
2  from MyCapytain.resources.texts.local.capitains.cts import CapitainsCtsText
3  from MyCapytain.common.constants import Mimetypes, XPATH_NAMESPACES
4  from lxml.etree import tostring
5
6  # We open a file
7  with open("./tests/testing_data/examples/text.martial.xml") as f:
8      # We initiate a Text object giving the IO instance to resource argument
9      text = CapitainsCtsText(resource=f)
10
11 # Text objects have a citation property
12 # len(Citation(...)) gives the depth of the citation scheme
13 # in the case of this sample, this would be 3 (Book, Poem, Line)
14 for ref in text.getReffs(level=len(text.citation)):
15     # We retrieve a Passage object for each reference that we find
16     # We can pass the reference many way, including in the form of a list of strings
17     # We use the _simple parameter to get a fairly simple object
18     # Simple makes a straight object that has only the targeted node inside of it
19     psg = text.getTextualNode(subreference=ref, simple=True)
20     # We print the passage from which we retrieve <note> nodes
21     print("\t".join([ref, psg.export(Mimetypes.PLAINTEXT, exclude=["tei:note"])]))
22
23 """
24 You'll print something like the following :
25
26     1.pr.1      Spero me secutum in libellis meis tale temperamen-
27     1.pr.2      tum, ut de illis queri non possit quisquis de se bene
28     1.pr.3      senserit, cum salva infimarum quoque personarum re-
29     1.pr.4      verentia ludant; quae adeo antiquis auctoribus defuit, ut
30     1.pr.5      nominibus non tantum veris abusi sint, sed et magnis.
31     1.pr.6      Mihi fama vilis constet et probetur in me novissimum
32
33 """
34
35 # It is possible that what you're interested in is a little more complex
36 # Like for example, getting a specific text sample with a specific reference
37 # In TEI !
38
39 # We open another such as Cicero's texts !
40 with open("./tests/testing_data/examples/text.cicero.xml") as f:
41     # We initiate a Text object giving the IO instance to resource argument
42     text = CapitainsCtsText(resource=f)
43     # We are specifically interested in the portion 28-30
44     # Note that we won't use 28-30 as cross passage reference won't work properly
45     p28_29 = text.getTextualNode("28-29")
46

```

(continues on next page)

(continued from previous page)

```

47  # And we want to be able to work with the xml
48  # To be injected in a third party API for lemmatization purposes
49  xml = p28_29.export(Mimetypes.XML.Std)
50  print("XML of 28-29")
51  print(xml)
52  print("-----")
53
54  # But what we really want to do, is suppress the note from the XML.
55  # So we export to an LXML Object
56  document = p28_29.export(Mimetypes.PYTHON.ETREE)
57  # We remove some XML
58  for element in document.xpath("//tei:note", namespaces=XPATH_NAMESPACES):
59      element.getparent().remove(element)
60  # And we print using LXML constants
61  print("Clean XML of 28-29")
62  print(tostring(document, encoding=str))
63  print("-----")
64

```

## 3.4 Known issues and recommendations

### 3.4.1 XPath Issues

*lxml*, which is the package powering xml support here, does not accept XPath notations such as `/div/(a or b)[@n]`. Solution for this edge case is `/div/*[self::a or self::b][@n]`

## 3.5 MyCapytain API Documentation

### 3.5.1 Utilities, metadata and references

Module `common` contains tools such as a namespace dictionary as well as cross-implementation objects, like URN, Citations...

#### Base

**class** `MyCapytain.common.base.Exportable` (\*args, \*\*kwargs)

Objects that supports Export

**Variables** `EXPORT_TO` – List of Mimetypes the resource can export to

**export** (*output=None*, \*\*kwargs)

Export the collection item in the Mimetype required.

**Parameters** `output` (*str*) – Mimetype to export to (Uses `MyCapytain.common.utils.Mimetypes`)

**Returns** Object using a different representation

**export\_capacities**

List Mimetypes that current object can export to

## Constants

`MyCapytain.common.constants.XPATH_NAMESPACES = {'cpt': 'http://purl.org/capitains/ns/1.0#`  
List of XPath Namespaces used in guidelines

`class MyCapytain.common.constants.RDF_NAMESPACES`  
Namespaces Constants used to provide Namespace capacities across the library

### Variables

- **CTS** – CTS Namespace
- **TEI** – TEI Namespace
- **DC** – DC Elements
- **CAPITAINS** – CapiTainS Ontology

`class MyCapytain.common.constants.Mimetypes`  
Mimetypes constants that are used to provide export functionality to base MyCapytain object.

### Variables

- **JSON** – JSON Resource mimetype
- **XML** – XML Resource mimetype
- **PYTHON** – Python Native Object
- **PLAINTEXT** – Plain string format

`class JSON`  
Json Mimetype

### Variables

- **Std** – Standard JSON Export
- **CTS** – CTS Json Export

`class DTS`  
JSON DTS Expression

### Variables

- **Std** – Standard DTS Json-LD Expression
- **NoParents** – DTS Json-LD Expression without parents expression

`class PYTHON`  
Python Native Objects

### Variables

- **NestedDict** – Nested Dictionary Object
- **ETREE** – Python LXML Etree Object

`class MyCapytain`  
MyCapytain Objects  
**Variables** **ReadableText** – MyCapytain.resources.prototypes.text.CtsText

`class XML`  
XML Mimetype

### Variables

- **Std** – Standard XML Export
- **RDF** – RDF XML Expression Export

- **CTS** – CTS API XML Expression Export
- **TEI** – TEI XML Expression Export

**class CapiTainS**  
CapiTainS Guideline XML Structured metadata

`MyCapytain.common.constants.bind_graph` (*graph=None*)  
Bind a graph with generic MyCapytain prefixes

**Parameters** `graph` – Graph (Optional)

**Returns** Bound graph

## URN, References and Citations

### MyCapytain Base Objects

**class** `MyCapytain.common.reference.NodeId` (*identifier=None, children=None, parent=None, siblings=(None, None), depth=None*)

Collection of directional references for a Tree

#### Parameters

- **identifier** (*str*) – Current object identifier
- **children** (*[str]*) – Current node Children’s Identifier
- **parent** (*str*) – Parent of the current node
- **siblings** (*str*) – Previous and next node of the current node
- **depth** (*int*) – Depth of the node in the global hierarchy of the text tree

**class** `MyCapytain.common.reference.BaseCitationSet` (*children=None*)  
A citation set is a collection of citations that optionnaly can be matched using a `.match()` function

**Parameters** `children` (*[BaseCitation]*) – List of Citation

**class** `MyCapytain.common.reference.BaseReference`  
BaseReference represents a passage identifier, range or not

It is made of two major properties : `.start` and `.end`

To check if the object is a range, you can use the method `.is_range()`

**class** `MyCapytain.common.reference.BaseReferenceSet`  
A BaseReferenceSet is a set of Reference (= a bag of identifier) that can carry citation and level information (what kind of reference is this reference ? Where am I in the levels of the current document ?)

It can be iterate like a tuple and has a `.citation` and `.level` property

### Canonical Text Services Objects

**class** `MyCapytain.common.reference.Citation` (*name=None, xpath=None, scope=None, refs-Decl=None, child=None*)

A citation object gives informations about the scheme

#### Parameters

- **name** (*basestring*) – Name of the citation (e.g. “book”)
- **xpath** (*basestring*) – Xpath of the citation (As described by CTS norm)

- **scope** – Scope of the citation (As described by CTS norm)
- **refsDecl** (*basestring*) – refsDecl version
- **child** (*Citation*) – A citation

#### Variables

- **name** – Name of the citation (e.g. “book”)
- **xpath** – Xpath of the citation (As described by CTS norm)
- **scope** – Scope of the citation (As described by CTS norm)
- **refsDecl** – refsDecl version
- **child** – A citation

**class** MyCapytain.common.reference.CtsReference

A reference object giving information

#### Example

```
>>> a = CtsReference("1.1@Achilles[1]-1.2@Zeus[1]")
>>> b = CtsReference("1.1")
>>> CtsReference("1.1-2.2.2").highest == CtsSinglePassageId("1.1")
```

Reference object supports the following magic methods : len(), str() and eq().

#### Example

```
>>> len(a) == 2 && len(b) == 1
>>> str(a) == "1.1@Achilles[1]-1.2@Zeus[1]"
>>> b == CtsReference("1.1") && b != a
```

**Note:** Reference(...).subreference and .list are not available for range. You will need to convert .start or .end to a Reference

```
>>> ref = CtsReference('1.2.3')
```

**class** MyCapytain.common.reference.CtsReferenceSet

A CTS version of the BaseReferenceSet

## Distributed Text Services Objects

**class** MyCapytain.common.reference.URN (*urn*)

A URN object giving all useful sections

**Parameters** *urn* (*str*) – A CTS URN

#### Variables

- **NAMESPACE** – Constant representing the URN until its namespace
- **TEXTGROUP** – Constant representing the URN until its textgroup
- **WORK** – Constant representing the URN until its work
- **VERSION** – Constant representing the URN until its version
- **PASSAGE** – Constant representing the URN until its full passage

- **PASSAGE\_START** – Constant representing the URN until its passage (end excluded)
- **PASSAGE\_END** – Constant representing the URN until its passage (start excluded)
- **NO\_PASSAGE** – Constant representing the URN until its passage excluding its passage
- **COMPLETE** – Constant representing the complete URN

**Example**

```
>>> a = URN(urn="urn:cts:latinLit:phi1294.phi002.perseus-lat2:1.1")
```

URN object supports the following magic methods : len(), str() and eq(), gt() and lt().

**Example**

```
>>> b = URN("urn:cts:latinLit:phi1294.phi002")
>>> a != b
>>> a > b # It has more member. Only member count is compared
>>> b < a
>>> len(a) == 5 # CtsReference is not counted to not induce count_
↳equivalencies with the optional version
>>> len(b) == 4
```

**class** MyCapytain.common.reference.DtsCitation (name=None, children=None, root=None)

**class** MyCapytain.common.reference.DtsCitationSet (children=None)  
Set of citations following the DTS model (Unlike CTS, one citation can have two or more children)

**Metadata containers**

**class** MyCapytain.common.metadata.Metadata (node=None, \*args, \*\*kwargs)  
A metadatum aggregation object provided to centralize metadata

**Parameters** **keys** ([text\_type]) – A metadata field names list

**Variables**

- **EXPORT\_TO** – List of exportable supported formats
- **DEFAULT\_EXPORT** – Default export (CTS XML Inventory)
- **STORE** – RDF Store

**add** (key, value, lang=None)  
Add a triple to the graph related to this node

**Parameters**

- **key** – Predicate of the triple
- **value** – Object of the triple
- **lang** – Language of the triple if applicable

**export** (output=None, \*\*kwargs)  
Export the collection item in the Mime type required.

**Parameters** **output** (str) – Mime type to export to (Uses MyCapytain.common.utils.Mimetypes)

**Returns** Object using a different representation

**export\_capacities**

List Mimetypees that current object can export to

**get** (*key, lang=None*)

Returns triple related to this node. Can filter on lang

**Parameters**

- **key** – Predicate of the triple
- **lang** – Language of the triple if applicable

**Return type** Literal or BNode or URIRef

**static getOr** (*subject, predicate, \*args, \*\*kwargs*)

Retrieve a metadata node or generate a new one

**Parameters**

- **subject** – Subject to which the metadata node should be connected
- **predicate** – Predicate by which the metadata node should be connected

**Returns** Metadata for given node

**Return type** Metadata

**get\_single** (*key, lang=None*)

Returns a single triple related to this node.

**Parameters**

- **key** – Predicate of the triple
- **lang** – Language of the triple if applicable

**Return type** Literal or BNode or URIRef

**graph**

Quick access to the graph this node is connected to

**Return type** Graph

**predicate\_object** (*predicate=None, obj=None*)

Retrieve predicate and object around this object

**Parameters**

- **predicate** – Predicate to match, None to match all
- **obj** – Object to match, None to match all

**Returns** List of resources

**remove** (*predicate=None, obj=None*)

Remove triple matching the predicate or the object

**Parameters**

- **predicate** – Predicate to match, None to match all
- **obj** – Object to match, None to match all

**set** (*key: rdflib.term.URIRef, value: Union[rdflib.term.Literal, rdflib.term.BNode, rdflib.term.URIRef, str, int], lang: Optional[str] = None*)

Set the VALUE for KEY predicate in the Metadata Graph

**Parameters**

- **key** – Predicate to be set (eg. DCT.creator)
- **value** – Value to be stored (eg. “Cicero”)
- **lang** – [Optional] Language of the value (eg. “la”)

**unlink** (*subj=None, predicate=None*)

Remove triple where Metadata is the object

**Parameters**

- **subj** – Subject to match, None to match all
- **predicate** – Predicate to match, None to match all

## Utilities

### 3.5.2 API Retrievers

Module endpoints contains prototypes and implementation of retrievers in MyCapytain

#### CTS 5 API

**class** MyCapytain.retrievers.cts5.**HttpCtsRetriever** (*endpoint, inventory=None*)

Bases: MyCapytain.retrievers.prototypes.CtsRetriever

Basic integration of the MyCapytain.retrievers.proto.CTS abstraction

**call** (*parameters*)

Call an endpoint given the parameters

**Parameters** **parameters** (*dict*) – Dictionary of parameters

**Return type** *text*

**getCapabilities** (*inventory=None, urn=None*)

Retrieve the inventory information of an API

**Parameters**

- **inventory** (*text*) – Name of the inventory
- **urn** (*str*) – URN to filter with

**Return type** *str*

**getFirstUrn** (*urn, inventory=None*)

Retrieve the first passage urn of a text

**Parameters**

- **urn** (*text*) – URN identifying the text
- **inventory** (*text*) – Name of the inventory

**Return type** *str*

**getLabel** (*urn, inventory=None*)

Retrieve informations about a CTS Urn

**Parameters**

- **urn** (*text*) – URN identifying the text’s passage (Minimum depth : 1)

- **inventory** (*text*) – Name of the inventory

**Return type** *str*

**getMetadata** (*objectId=None, \*\*filters*)

Request metadata about a text or a collection

**Parameters**

- **objectId** – Filter for some object identifier
- **filters** – Kwarg parameters. URN and Inv are available

**Returns** GetCapabilities CTS API request response

**getPassage** (*urn, inventory=None, context=None*)

Retrieve a passage

**Parameters**

- **urn** (*text*) – URN identifying the text's passage (Minimum depth : 1)
- **inventory** (*text*) – Name of the inventory
- **context** (*int*) – Number of citation units at the same level of the citation hierarchy as the requested urn, immediately preceding and immediately following the requested urn to include in the reply

**Return type** *str*

**getPassagePlus** (*urn, inventory=None, context=None*)

Retrieve a passage and information about it

**Parameters**

- **urn** (*text*) – URN identifying the text's passage (Minimum depth : 1)
- **inventory** (*text*) – Name of the inventory
- **context** (*int*) – Number of citation units at the same level of the citation hierarchy as the requested urn, immediately preceding and immediately following the requested urn to include in the reply

**Return type** *str*

**getPrevNextUrn** (*urn, inventory=None*)

Retrieve the previous and next passage urn of one passage

**Parameters**

- **urn** (*text*) – URN identifying the text's passage (Minimum depth : 1)
- **inventory** (*text*) – Name of the inventory

**Return type** *str*

**getReffs** (*textId, level=1, subreference=None*)

Retrieve the siblings of a textual node

**Parameters**

- **textId** (*str*) – CtsTextMetadata Identifier
- **level** (*int*) – Depth for retrieval
- **subreference** (*str*) – CapitainsCtsPassage Reference

**Returns** List of references

**Return type** `[str]`

**getSiblings** (*textId, subreference*)

Retrieve the siblings of a textual node

**Parameters**

- **textId** – CtsTextMetadata Identifier
- **reference** – CapitainsCtsPassage Reference

**Returns** GetPrevNextUrn request response from the endpoint

**getTextualNode** (*textId, subreference=None, prevnext=False, metadata=False*)

Retrieve a text node from the API

**Parameters**

- **textId** – CtsTextMetadata Identifier
- **subreference** – CapitainsCtsPassage Reference
- **prevnext** – Retrieve graph representing previous and next passage
- **metadata** – Retrieve metadata about the passage and the text

**Returns** GetPassage or GetPassagePlus CTS API request response

**getValidReff** (*urn, inventory=None, level=None*)

Retrieve valid urn-references for a text

**Parameters**

- **urn** (*text*) – URN identifying the text
- **inventory** (*text*) – Name of the inventory
- **level** (*int*) – Depth of references expected

**Returns** XML Response from the API as string

**Return type** `str`

## Prototypes

**class** `MyCapytain.retrievers.prototypes.API` (*endpoint*)

Bases: `object`

API Prototype object

**Parameters**

- **self** (*API*) – Object
- **endpoint** (*text*) – URL of the API

**Variables** `endpoint` – Url of the endpoint

**class** `MyCapytain.retrievers.prototypes.CitableTextServiceRetriever` (*endpoint*)

Bases: `MyCapytain.retrievers.prototypes.API`

Citable CtsTextMetadata Service retrievers should have at least have some of the following properties

**getMetadata** (*objectId=None, \*\*filters*)

Request metadata about a text or a collection

**Parameters**

- **objectId** – CtsTextMetadata Identifier
- **filters** – Kwarg parameters. URN and Inv are available

**Returns** Metadata of text from an API or the likes as bytes

**getReffs** (*textId*, *level=1*, *subreference=None*)

Retrieve the siblings of a textual node

**Parameters**

- **textId** (*str*) – CtsTextMetadata Identifier
- **level** (*int*) – Depth for retrieval
- **subreference** (*str*) – CapitainsCtsPassage Reference

**Returns** List of references

**Return type** `[str]`

**getSiblings** (*textId*, *subreference*)

Retrieve the siblings of a textual node

**Parameters**

- **textId** – CtsTextMetadata Identifier
- **subreference** – CapitainsCtsPassage Reference

**Returns** Siblings references from an API or the likes as bytes

**getTextualNode** (*textId*, *subreference=None*, *prevnext=False*, *metadata=False*)

Retrieve a text node from the API

**Parameters**

- **textId** – CtsTextMetadata Identifier
- **subreference** – CapitainsCtsPassage Reference
- **prevnext** – Retrieve graph representing previous and next passage
- **metadata** – Retrieve metadata about the passage and the text

**Returns** CtsTextMetadata of a CapitainsCtsPassage from an API or the likes as bytes

**class** MyCapytain.retrieveers.prototypes.**CtsRetriever** (*endpoint*)

Bases: MyCapytain.retrieveers.prototypes.CitableTextServiceRetriever

CTS API Endpoint Prototype

**getCapabilities** (*inventory*)

Retrieve the inventory information of an API

**Parameters** **inventory** (*text*) – Name of the inventory

**Return type** `str`

**getFirstUrn** (*urn*, *inventory*)

Retrieve the first passage urn of a text

**Parameters**

- **urn** (*text*) – URN identifying the text
- **inventory** (*text*) – Name of the inventory

**Return type** `str`

**getLabel** (*urn, inventory*)

Retrieve informations about a CTS Urn

**Parameters**

- **urn** (*text*) – URN identifying the text’s passage (Minimum depth : 1)
- **inventory** (*text*) – Name of the inventory

**Return type** *str*

**getMetadata** (*objectId=None, \*\*filters*)

Request metadata about a text or a collection

**Parameters**

- **objectId** – CtsTextMetadata Identifier
- **filters** – Kwargs parameters. URN and Inv are available

**Returns** Metadata of text from an API or the likes as bytes

**getPassage** (*urn, inventory, context=None*)

Retrieve a passage

**Parameters**

- **urn** (*text*) – URN identifying the text’s passage (Minimum depth : 1)
- **inventory** (*text*) – Name of the inventory
- **context** (*int*) – Number of citation units at the same level of the citation hierarchy as the requested urn, immediately preceding and immediately following the requested urn to include in the reply

**Return type** *str*

**getPassagePlus** (*urn, inventory, context=None*)

Retrieve a passage and informations about it

**Parameters**

- **urn** (*text*) – URN identifying the text’s passage (Minimum depth : 1)
- **inventory** (*text*) – Name of the inventory
- **context** (*int*) – Number of citation units at the same level of the citation hierarchy as the requested urn, immediately preceding and immediately following the requested urn to include in the reply

**Return type** *str*

**getPrevNextUrn** (*urn, inventory*)

Retrieve the previous and next passage urn of one passage

**Parameters**

- **urn** (*text*) – URN identifying the text’s passage (Minimum depth : 1)
- **inventory** (*text*) – Name of the inventory

**Return type** *str*

**getReffs** (*textId, level=1, subreference=None*)

Retrieve the siblings of a textual node

**Parameters**

- **textId** (*str*) – CtsTextMetadata Identifier
- **level** (*int*) – Depth for retrieval
- **subreference** (*str*) – CapitainsCtsPassage Reference

**Returns** List of references

**Return type** [*str*]

**getSiblings** (*textId, subreference*)

Retrieve the siblings of a textual node

**Parameters**

- **textId** – CtsTextMetadata Identifier
- **subreference** – CapitainsCtsPassage Reference

**Returns** Siblings references from an API or the likes as bytes

**getTextualNode** (*textId, subreference=None, prevnext=False, metadata=False*)

Retrieve a text node from the API

**Parameters**

- **textId** – CtsTextMetadata Identifier
- **subreference** – CapitainsCtsPassage Reference
- **prevnext** – Retrieve graph representing previous and next passage
- **metadata** – Retrieve metadata about the passage and the text

**Returns** CtsTextMetadata of a CapitainsCtsPassage from an API or the likes as bytes

**getValidReff** (*urn, inventory, level=1*)

Retrieve valid urn-references for a text

**Parameters**

- **urn** (*text*) – URN identifying the text
- **inventory** (*text*) – Name of the inventory
- **level** (*int*) – Depth of references expected

**Return type** *str*

### 3.5.3 Resolvers

#### Remote CTS API

**class** MyCapytain.resolvers.cts.api.**HttpCtsResolver** (*endpoint*)

HttpCtsResolver provide a resolver for CTS API http endpoint.

**Parameters** **endpoint** (*HttpCtsRetriever*) – CTS API Retriever

**Variables** **endpoint** – CTS API Retriever

**endpoint**

CTS Endpoint of the resolver

**Returns** CTS Endpoint

**Return type** HttpCtsRetriever

**getMetadata** (*objectId=None, \*\*filters*)

Request metadata about a text or a collection

**Parameters**

- **objectId** (*str*) – Object Identifier to filter on
- **filters** (*dict*) – Kwarg parameters.

**Returns** Collection

**getReffs** (*textId, level=1, subreference=None*)

Retrieve the siblings of a textual node

**Parameters**

- **textId** (*str*) – CtsTextMetadata Identifier
- **level** (*int*) – Depth for retrieval
- **subreference** (*str*) – CapitainsCtsPassage Reference

**Returns** List of references

**Return type** [str]

**getSiblings** (*textId, subreference*)

Retrieve the siblings of a textual node

**Parameters**

- **textId** (*str*) – CtsTextMetadata Identifier
- **subreference** (*str*) – CapitainsCtsPassage Reference

**Returns** Tuple of references

**Return type** (str, str)

**getTextualNode** (*textId, subreference=None, prevnext=False, metadata=False*)

Retrieve a text node from the API

**Parameters**

- **textId** (*str*) – CtsTextMetadata Identifier
- **subreference** (*str*) – CapitainsCtsPassage Reference
- **prevnext** (*boolean*) – Retrieve graph representing previous and next passage
- **metadata** (*boolean*) – Retrieve metadata about the passage and the text

**Returns** CapitainsCtsPassage

**Return type** *CapitainsCtsPassage*

## Local CapiTainS Guidelines CTS Resolver

```
class MyCapytain.resolvers.cts.local.CtsCapitainsLocalResolver (resource,  
                                                                name=None,  
                                                                log-  
                                                                ger=None, dis-  
                                                                patcher=None,  
                                                                au-  
                                                                toparse=True)
```

XML Folder Based resolver. CtsTextMetadata and metadata resolver based on local directories

**Parameters**

- **resource** (*[str]*) – Resource should be a list of folders retaining data as Capitains Guidelines Repositories
- **name** (*str*) – Key used to differentiate Repository and thus enabling different repo to be used
- **logger** (*logging*) – Logging object

**Variables**

- **TEXT\_CLASS** – CtsTextMetadata Class [not instantiated] to be used to parse Texts. Can be changed to support Cache for example
- **DEFAULT\_PAGE** – Default Page to show
- **PER\_PAGE** – Tuple representing the minimal number of texts returned, the default number and the maximum number of texts returned

**getMetadata** (*objectId=None, \*\*filters*)

Request metadata about a text or a collection

**Parameters**

- **objectId** (*str*) – Object Identifier to filter on
- **filters** (*dict*) – Kwargs parameters.

**Returns** Collection

**getReffs** (*textId, level=1, subreference=None*)

Retrieve the siblings of a textual node

**Parameters**

- **textId** (*str*) – CtsTextMetadata Identifier
- **level** (*int*) – Depth for retrieval
- **subreference** (*str*) – CapitainsCtsPassage CtsReference

**Returns** List of references

**Return type** [*str*]

**getSiblings** (*textId, subreference: MyCapytain.common.reference.\_capitains\_cts.CtsReference*)

Retrieve the siblings of a textual node

**Parameters**

- **textId** (*str*) – CtsTextMetadata Identifier
- **subreference** (*str*) – CapitainsCtsPassage CtsReference

**Returns** Tuple of references

**Return type** (*str, str*)

**getTextualNode** (*textId, subreference=None, prevnext=False, metadata=False*)

Retrieve a text node from the API

**Parameters**

- **textId** (*str*) – CtsTextMetadata Identifier
- **subreference** (*str*) – CapitainsCtsPassage CtsReference
- **prevnext** (*boolean*) – Retrieve graph representing previous and next passage

- **metadata** (*boolean*) – Retrieve metadata about the passage and the text

**Returns** CapitainsCtsPassage

**Return type** *CapitainsCtsPassage*

**static pagination** (*page, limit, length*)

Help for pagination :param page: Provided Page :param limit: Number of item to show :param length: Length of the list to paginate :return: (Start Index, End Index, Page Number, Item Count)

**parse** (*resource*)

Parse a list of directories and reads it into a collection

**Parameters** **resource** – List of folders

**Returns** An inventory resource and a list of CtsTextMetadata metadata-objects

**read** (*identifier, path*)

Retrieve and parse a text given an identifier

**Parameters**

- **identifier** (*str*) – Identifier of the text
- **path** (*str*) – Path of the text

**Returns** Parsed Text

**Return type** *CapitainsCtsText*

**xmlparse** (*file*)

Parse a XML file :param file: Opened File :return: Tree

## Dispatcher

**class** MyCapytain.resolvers.utils.**CollectionDispatcher** (*collection, default\_inventory\_name=None*)

Collection Dispatcher provides a utility to divide automatically texts and collections into different collections

**Parameters**

- **collection** – The root collection
- **default\_inventory\_name** – The default name of the default collection

**add** (*func, inventory\_name*)

Register given function as a filter.

If this function “func” returns True when given an object, said object will be dispatched to Collection(inventory\_name)

**Parameters**

- **func** – Callable
- **inventory\_name** – Identifier of the collection to dispatch to

**dispatch** (*collection, \*\*kwargs*)

Dispatch a collection using internal filters

**Parameters**

- **collection** – Collection object
- **kwargs** – Additional keyword arguments that could be used by internal filters

**Returns** None

**Raises**

**inventory** (*inventory\_name*)

Decorator to register filters for given inventory. For a function “abc”, it has the same effect

**Parameters** *inventory\_name* –

**Returns**

```
tic = CtsTextInventoryCollection()
latin = CtsTextInventoryMetadata("urn:perseus:latinLit", parent=tic)
latin.set_label("Classical Latin", "eng")
dispatcher = CollectionDispatcher(tic)

@dispatcher.inventory("urn:perseus:latinLit")
def dispatchLatinLit(collection, path=None, **kwargs):
    if collection.id.startswith("urn:cts:latinLit:"):
        return True
    return False
```

**methods**

List of methods to dispatch resources.

**Each element is a tuple with two elements :**

- First one is the inventory identifier to dispatch to
- Second one is a function which, if returns true, will activate dispatching to given

**Return type** List

## Prototypes

**class** MyCapytain.resolvers.prototypes.Resolver

Resolver provide a native python API which returns python objects.

Initiation of resolvers are dependent on the implementation of the prototype

**getMetadata** (*objectId: str = None, \*\*filters*) → MyCapytain.resources.prototypes.metadata.Collection  
Request metadata about a text or a collection

**Parameters**

- **objectId** (*str*) – Object Identifier to filter on
- **filters** (*dict*) – Kwargs parameters.

**Returns** Collection

**getReffs** (*textId: str, level: int = 1, subreference: Union[str, MyCapytain.common.reference.\_base.BaseReference] = None, include\_descendants: bool = False, additional\_parameters: Optional[Dict[str, Any]] = None*) → MyCapytain.common.reference.\_base.BaseReferenceSet  
Retrieve the siblings of a textual node

**Parameters**

- **textId** (*str*) – CtsTextMetadata Identifier
- **level** (*int*) – Depth for retrieval

- **subreference** (*str*) – CapitainsCtsPassage Reference
- **include\_descendants** –
- **additional\_parameters** –

**Returns** List of references

**Return type** [*str*]

..toDo :: This starts to be a bloated function. ...

```

getSiblings (textId:          str,          subreference:          Union[str,          MyCapy-
                tain.common.reference._base.BaseReference])          →          Tu-
                ple[MyCapytain.common.reference._base.BaseReference,          MyCapy-
                tain.common.reference._base.BaseReference]

```

Retrieve the siblings of a textual node

**Parameters**

- **textId** (*str*) – CtsTextMetadata Identifier
- **subreference** (*str*) – CapitainsCtsPassage Reference

**Returns** Tuple of references

**Return type** (*str*, *str*)

```

getTextualNode (textId:          str,          subreference:          Union[str,          MyCapy-
                tain.common.reference._base.BaseReference] = None, prevnext: bool = False,
                metadata: bool = False) → MyCapytain.resources.prototypes.text.TextualNode

```

Retrieve a text node from the API

**Parameters**

- **textId** (*str*) – CtsTextMetadata Identifier
- **subreference** (*str*) – CapitainsCtsPassage Reference
- **prevnext** (*boolean*) – Retrieve graph representing previous and next passage
- **metadata** (*boolean*) – Retrieve metadata about the passage and the text

**Returns** CapitainsCtsPassage

**Return type** *CapitainsCtsPassage*

### 3.5.4 Texts and inventories

#### Text

#### TEI based texts

#### Locally read text

```

class MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText (urn=None,
                                                                    cita-
                                                                    tion=None,
                                                                    re-
                                                                    source=None)
    Bases: MyCapytain.resources.texts.local.capitains.cts._SharedMethods,
           MyCapytain.resources.prototypes.cts.text.PrototypeCtsText

```

Implementation of CTS tools for local files

#### Parameters

- **urn** (*MyCapytain.common.reference.\_capitains\_cts.URN*) – A URN identifier
- **resource** (*lxml.etree.\_Element*) – A resource
- **citation** (*Citation*) – Highest XmlCtsCitation level
- **autoreffs** (*bool*) – Parse references on load (default : True)

Variables **resource** – lxml

**DEFAULT\_EXPORT** = 'text/plain'

**EXPORT\_TO** = ['python/lxml', 'text/xml', 'python/NestedDict', 'text/plain', 'text/xml:t

**PLAINTEXT\_STRING\_JOIN** = ' '

**asNode** () → rdflib.term.Identifier

#### childIds

Identifiers of children

**Returns** Identifiers of children

#### children

Children TextualNode

#### citation

Citation system of the object

**Return type** *Citation*

**default\_exclude** = []

#### depth

Depth of the node in the global hierarchy of the text tree

**export** (*output: str = None, exclude: List[str] = None, \*\*kwargs*)

Export the collection item in the Mime type required.

..note:: If current implementation does not have special mimetypes, reuses default\_export method

#### Parameters

- **output** (*str*) – Mime type to export to (Uses *MyCapytain.common.constants.Mimetypes*)
- **exclude** (*[str]*) – Information to exclude. Specific to implementations

**Returns** Object using a different representation

#### export\_capacities

List Mime types that current object can export to

#### first

First TextualNode

#### firstId

First child's id of current TextualNode

**getLabel** () → *MyCapytain.resources.prototypes.metadata.Collection*

Retrieve metadata about the text

**Return type** *Collection*

**Returns** Retrieve Label informations in a Collection format

**getReffs** (*level: int = 1, subreference: MyCapytain.common.reference.\_capitains\_cts.CtsReference = None*) → MyCapytain.common.reference.\_capitains\_cts.CtsReferenceSet  
CtsReference available at a given level

**Parameters**

- **level** – Depth required. If not set, should retrieve first encountered level (1 based)
- **subreference** – Subreference (optional)

**Returns** List of levels

**getTextualNode** (*subreference=None, simple=False*)  
Finds a passage in the current text

**Parameters**

- **subreference** (*Union[list, CtsReference]*) – Identifier of the subreference / passages
- **simple** (*boolean*) – If set to true, retrieves nodes up to the given one, cleaning non required siblings.

**Return type** *CapitainsCtsPassage*, *ContextPassage*

**Returns** Asked passage

**getValidReff** (*level: int = 1, reference: MyCapytain.common.reference.\_capitains\_cts.CtsReference = None, \_debug: bool = False*) → MyCapytain.common.reference.\_capitains\_cts.CtsReferenceSet  
Retrieve valid passages directly

**Parameters**

- **level** (*int*) – Depth required. If not set, should retrieve first encountered level (1 based)
- **reference** (*CtsReference*) – CapitainsCtsPassage Reference
- **\_debug** (*bool*) – Check on passages duplicates

**Returns** List of levels

---

**Note:** GetValidReff works for now as a loop using CapitainsCtsPassage, subinstances of CtsTextMetadata, to retrieve the valid informations. Maybe something is more powerfull ?

---

**get\_creator** (*lang: str = None*) → rdflib.term.Literal  
Get the DC Creator literal value

**Parameters** **lang** – Language to retrieve

**Returns** Creator string representation

**Return type** Literal

**get\_cts\_metadata** (*key: str, lang: str = None*) → rdflib.term.Literal  
Get easily a metadata from the CTS namespace

**Parameters**

- **key** – CTS property to retrieve
- **lang** – Language in which it should be

**Returns** Literal value of the CTS graph property

**get\_description** (*lang: str = None*) → rdflib.term.Literal

Get the description of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Description string representation

**Return type** Literal

**get\_subject** (*lang=None*) → rdflib.term.Literal

Get the subject of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Subject string representation

**Return type** Literal

**get\_title** (*lang: str = None*) → rdflib.term.Literal

Get the title of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Title string representation

**Return type** Literal

**graph**

**id**

Identifier of the text

**Returns** Identifier of the text

**last**

Last CapitainsCtsPassage

**lastId**

Last child's id of current TextualNode

**metadata**

Metadata information about the text

**Returns** Collection object with metadata about the text

**Return type** Metadata

**next**

Get Next TextualNode

**nextId**

Next Id

**parent**

Parent TextualNode

**parentId**

Parent Id

**plaintext\_string\_join**

String used to join xml node's texts in export

**prev**

Get Previous TextualNode

**prevId**

Previous Id (Sibling)

**reffs**

Get all valid reffs for every part of the CtsText

**Return type** [str]

**set\_creator** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)

Set the DC Creator literal value

**Parameters**

- **value** – Value of the creator node
- **lang** – Language in which the value is

**set\_description** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)

Set the DC Description literal value

**Parameters**

- **value** – Value of the title node
- **lang** – Language in which the value is

**set\_metadata\_from\_collection** (*text\_metadata: MyCapytain.resources.prototypes.cts.inventory.CtsTextMetadata*)

Set the object metadata using its collections recursively

**Parameters** **text\_metadata** (*CtsTextMetadata*) – Object representing the current text as a collection

**set\_subject** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)

Set the DC Subject literal value

**Parameters**

- **value** – Value of the subject node
- **lang** – Language in which the value is

**set\_title** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)

Set the DC Title literal value

**Parameters**

- **value** – Value of the title node
- **lang** – Language in which the value is

**siblingsId**

Siblings Id

**test** ()

Parse the object and generate the children

**text**

String representation of the text

**Returns** String representation of the text

**textObject**

Textual Object with full capacities (Unlike Simple CapitainsCtsPassage)

**Return type** CtsTextMetadata, *CapitainsCtsPassage*

**Returns** Textual Object with full capacities (Unlike Simple CapitainsCtsPassage)

**tostring** (*\*args, \*\*kwargs*)

Transform the CapitainsCtsPassage in XML string

**Parameters**

- **args** – Ordered arguments for `etree.tostring()` (except the first one)
- **kwargs** – Named arguments

**Returns****urn**

URN Identifier of the object

**xml**XML Representation of the `CapitainsCtsPassage`**Return type** `lxml.etree._Element`**Returns** XML element representing the passage**xpath** (*\*args*, *\*\*kwargs*)

Perform XPath on the passage XML

**Parameters**

- **args** – Ordered arguments for `etree._Element().xpath()`
- **kwargs** – Named arguments

**Returns** Result list**Return type** `list(etree._Element)`

**class** `MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage` (*reference*,  
*urn=None*,  
*ci-*  
*ta-*  
*tion=None*,  
*re-*  
*source=None*,  
*text=None*)

**Bases:** `MyCapytain.resources.texts.local.capitains.cts._SharedMethods`,  
`MyCapytain.resources.prototypes.cts.text.PrototypeCtsPassage`

`CapitainsCtsPassage` class for local texts which rebuilds the tree up to the passage.

For design purposes, some people would prefer the output of `GetPassage` to be consistent. `ContextPassage` rebuilds the tree of the text up to the passage, keeping attributes of original nodes

**Example** : for a text with a citation scheme with following `refsDecl` :  
`/TEI/text/body/div[@type='edition']/div[@n='$1']/div[@n='$2']/l[@n='$3']` and a passage 1.1.1-1.2.3,  
this class will build an XML tree looking like the following

```
<TEI ...>
  <text ...>
    <body ...>
      <div type='edition' ...>
        <div n='1' ...>

          <div n='1' ...>
            <l n='1'>...</l>
            ...
          </div>
          <div n='2' ...>
            <l n='3'>...</l>
```

(continues on next page)

(continued from previous page)

```

        </div>
    </div>
</div>
</body>
</text>
</TEI>

```

**Parameters**

- **reference** (*CtsReference*) – CapitainsCtsPassage reference
- **urn** (URN) – URN of the source text or of the passage
- **citation** (*XmlCtsCitation*) – XmlCtsCitation scheme of the text
- **resource** (*etree.\_Element*) – Element representing the passage
- **text** (*CtsTextMetadata*) – CtsTextMetadata containing the passage

---

**Note:** `.prev`, `.next`, `.first` and `.last` won't run on passage with a range made of two different level, such as 1.1-1.2.3 or 1-a.b. Those will raise *InvalidSiblingRequest*

---

**DEFAULT\_EXPORT** = 'text/plain'

**EXPORT\_TO** = ['python/lxml', 'text/xml', 'python/NestedDict', 'text/plain', 'text/xml:t

**PLAINTEXT\_STRING\_JOIN** = ' '

**asNode** () → rdflib.term.Identifier

**childIds**

Children of the passage

**Return type** *None, CtsReference*

**Returns** Dictionary of children, where key are subreferences

**children**

Children TextualNode

**citation**

Citation system of the object

**Return type** *Citation*

**default\_exclude** = []

**depth**

Depth of the node in the global hierarchy of the text tree

**export** (*output: str = None, exclude: List[str] = None, \*\*kwargs*)

Export the collection item in the Mimetype required.

..note:: If current implementation does not have special mimetypes, reuses default\_export method

**Parameters**

- **output** (*str*) – Mimetype to export to (Uses MyCapytain.common.constants.Mimetypes)
- **exclude** (*[str]*) – Information to exclude. Specific to implementations

**Returns** Object using a different representation

**export\_capacities**

List Mimetypes that current object can export to

**first**

First TextualNode

**firstId**

First child's id of current TextualNode

**getLabel** () → MyCapytain.resources.prototypes.metadata.Collection

Retrieve metadata about the text

**Return type** Collection

**Returns** Retrieve Label informations in a Collection format

**getReffs** (*level: int = 1, subreference: MyCapytain.common.reference.\_capitains\_cts.CtsReference = None*) → MyCapytain.common.reference.\_capitains\_cts.CtsReferenceSet  
CtsReference available at a given level

**Parameters**

- **level** – Depth required. If not set, should retrieve first encountered level (1 based)
- **subreference** – Subreference (optional)

**Returns** List of levels

**getTextualNode** (*subreference=None, \*args, \*\*kwargs*)

Finds a passage in the current text

**Parameters**

- **subreference** (*Union[list, CtsReference]*) – Identifier of the subreference / passages
- **simple** (*boolean*) – If set to true, retrieves nodes up to the given one, cleaning non required siblings.

**Return type** *CapitainsCtsPassage*, ContextPassage

**Returns** Asked passage

**getValidReff** (*level: int = 1, reference: MyCapytain.common.reference.\_capitains\_cts.CtsReference = None, \_debug: bool = False*) → MyCapytain.common.reference.\_capitains\_cts.CtsReferenceSet

Retrieve valid passages directly

**Parameters**

- **level** (*int*) – Depth required. If not set, should retrieve first encountered level (1 based)
- **reference** (*CtsReference*) – CapitainsCtsPassage Reference
- **\_debug** (*bool*) – Check on passages duplicates

**Returns** List of levels

---

**Note:** GetValidReff works for now as a loop using CapitainsCtsPassage, subinstances of CtsTextMetadata, to retrieve the valid informations. Maybe something is more powerfull ?

---

**get\_creator** (*lang: str = None*) → rdflib.term.Literal

Get the DC Creator literal value

**Parameters** `lang` – Language to retrieve

**Returns** Creator string representation

**Return type** Literal

**get\_cts\_metadata** (*key: str, lang: str = None*) → rdflib.term.Literal

Get easily a metadata from the CTS namespace

**Parameters**

- **key** – CTS property to retrieve
- **lang** – Language in which it should be

**Returns** Literal value of the CTS graph property

**get\_description** (*lang: str = None*) → rdflib.term.Literal

Get the description of the object

**Parameters** `lang` – Lang to retrieve

**Returns** Description string representation

**Return type** Literal

**get\_subject** (*lang=None*) → rdflib.term.Literal

Get the subject of the object

**Parameters** `lang` – Lang to retrieve

**Returns** Subject string representation

**Return type** Literal

**get\_title** (*lang: str = None*) → rdflib.term.Literal

Get the title of the object

**Parameters** `lang` – Lang to retrieve

**Returns** Title string representation

**Return type** Literal

**graph**

**id**

Identifier of the text

**Returns** Identifier of the text

**last**

Last CapitainsCtsPassage

**lastId**

Last child's id of current TextualNode

**metadata**

Metadata information about the text

**Returns** Collection object with metadata about the text

**Return type** Metadata

**next**

Next CapitainsCtsPassage (Interactive CapitainsCtsPassage)

**nextId**

Next passage

**Returns** Next passage at same level

**Return type** *None*, *CtsReference*

**parent**

Parent TextualNode

**parentId**

Parent Id

**plaintext\_string\_join**

String used to join xml node's texts in export

**prev**

Previous CapitainsCtsPassage (Interactive CapitainsCtsPassage)

**prevId**

Get the Previous passage reference

**Returns** Previous passage reference at the same level

**Return type** *None*, *CtsReference*

**reference**

CtsReference of the object

**set\_creator** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str]*, *lang: str = None*)

Set the DC Creator literal value

**Parameters**

- **value** – Value of the creator node
- **lang** – Language in which the value is

**set\_description** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str]*, *lang: str = None*)

Set the DC Description literal value

**Parameters**

- **value** – Value of the title node
- **lang** – Language in which the value is

**set\_metadata\_from\_collection** (*text\_metadata: MyCapytain.resources.prototypes.cts.inventory.CtsTextMetadata*)

Set the object metadata using its collections recursively

**Parameters** **text\_metadata** (*CtsTextMetadata*) – Object representing the current text as a collection

**set\_subject** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str]*, *lang: str = None*)

Set the DC Subject literal value

**Parameters**

- **value** – Value of the subject node
- **lang** – Language in which the value is

**set\_title** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str]*, *lang: str = None*)

Set the DC Title literal value

**Parameters**

- **value** – Value of the title node
- **lang** – Language in which the value is

**siblingsId**

Siblings Identifiers of the passage

**Return type** (str, str)

**text**

String representation of the text

**Returns** String representation of the text

**textObject**

CtsTextMetadata Object. Required for NextPrev

**Return type** *CapitainsCtsText*

**tostring** (\*args, \*\*kwargs)

Transform the CapitainsCtsPassage in XML string

**Parameters**

- **args** – Ordered arguments for etree.tostring() (except the first one)
- **kwargs** – Named arguments

**Returns**

**urn**

URN Identifier of the object

**xml**

XML Representation of the CapitainsCtsPassage

**Return type** lxml.etree.\_Element

**Returns** XML element representing the passage

**xpath** (\*args, \*\*kwargs)

Perform XPath on the passage XML

**Parameters**

- **args** – Ordered arguments for etree.\_Element().xpath()
- **kwargs** – Named arguments

**Returns** Result list

**Return type** list(etree.\_Element)

## CTS API Texts

Formerly MyCapytain.resources.texts.api (< 2.0.0)

**class** MyCapytain.resources.texts.remote.cts.CtsText (urn, retriever, citation=None, \*\*kwargs)

Bases: MyCapytain.resources.texts.remote.cts.\_SharedMethod, MyCapytain.resources.prototypes.cts.text.PrototypeCtsText

API CtsTextMetadata object

**Parameters**

- **urn** (*Union[URN, str, unicode]*) – A URN identifier
- **resource** (*CitableTextServiceRetriever*) – An API endpoint
- **citation** (*XmlCtsCitation*) – XmlCtsCitation for children level
- **id** (*List*) – Identifier of the subreference without URN informations

**DEFAULT\_EXPORT** = None

**DEFAULT\_LANG** = 'eng'

**EXPORT\_TO** = []

**asNode** () → rdflib.term.Identifier

**childIds**

Identifiers of children

**Returns** Identifiers of children

**children**

Children TextualNode

**citation**

Citation system of the object

**Return type** *Citation*

**default\_exclude** = []

**depth**

Depth of the current object

**Returns** Int representation of the depth based on URN information

**Return type** int

**export** (*output='text/plain', exclude=None, \*\*kwargs*)

Export the collection item in the Mimetype required.

..note:: If current implementation does not have special mimetypes, reuses default\_export method

**Parameters**

- **output** (*str*) – Mimetype to export to (Uses Mimetypes)
- **exclude** (*[str]*) – Informations to exclude. Specific to implementations

**Returns** Object using a different representation

**export\_capacities**

List Mimetypes that current object can export to

**first**

First TextualNode

**firstId**

Children passage

**Return type** str

**Returns** First children of the graph. Shortcut to self.graph.children[0]

**static firstUrn** (*resource*)

Parse a resource to get the first URN

**Parameters** **resource** (*etree.\_Element*) – XML Resource

**Returns** Tuple representing previous and next urn

**Return type** *str*

**getFirstUrn** (*reference=None*)

Get the first children URN for a given resource

**Parameters** **reference** (*CtsReference*, *str*) – CtsReference from which to find child  
(If None, find first reference)

**Returns** Children URN

**Return type** *URN*

**getLabel** ()

Retrieve metadata about the text

**Return type** Metadata

**Returns** Dictionary with label informations

**getPassagePlus** (*reference=None*)

Retrieve a passage and informations around it and store it in the object

**Parameters** **reference** (*CtsReference* or *List of text\_type*) – Reference of  
the passage

**Return type** *CtsPassage*

**Returns** Object representing the passage

**Raises** *TypeError* when reference is not a list or a Reference

**getPrevNextUrn** (*reference*)

Get the previous URN of a reference of the text

**Parameters** **reference** (*Union[CtsReference, str]*) – CtsReference from which to  
find siblings

**Returns** (Previous CapitainsCtsPassage CtsReference, Next CapitainsCtsPassage CtsReference)

**getReffs** (*level=1, subreference=None*)

Reference available at a given level

**Parameters**

- **level** (*Int*) – Depth required. If not set, should retrieve first encountered level (1 based)
- **subreference** (*CtsReference*) – Subreference (optional)

**Return type** [text\_type]

**Returns** List of levels

**getTextualNode** (*subreference=None*)

Retrieve a passage and store it in the object

**Parameters** **subreference** (*Union[CtsReference, URN, str, list]*) – CtsRef-  
erence of the passage (Note : if given a list, this should be a list of string that compose the  
reference)

**Return type** *CtsPassage*

**Returns** Object representing the passage

**Raises** *TypeError* when reference is not a list or a CtsReference

**getValidReff** (*level=1, reference=None*)

Given a resource, CtsText will compute valid reffs

**Parameters**

- **level** (*Int*) – Depth required. If not set, should retrieve first encountered level (1 based)
- **reference** (*CtsReference*) – CapitainsCtsPassage reference

**Return type** *list(str)*

**Returns** List of levels

**get\_creator** (*lang: str = None*) → *rdflib.term.Literal*

Get the DC Creator literal value

**Parameters** **lang** – Language to retrieve

**Returns** Creator string representation

**Return type** *Literal*

**get\_cts\_metadata** (*key: str, lang: str = None*) → *rdflib.term.Literal*

Get easily a metadata from the CTS namespace

**Parameters**

- **key** – CTS property to retrieve
- **lang** – Language in which it should be

**Returns** Literal value of the CTS graph property

**get\_description** (*lang: str = None*) → *rdflib.term.Literal*

Get the description of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Description string representation

**Return type** *Literal*

**get\_subject** (*lang=None*) → *rdflib.term.Literal*

Get the subject of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Subject string representation

**Return type** *Literal*

**get\_title** (*lang: str = None*) → *rdflib.term.Literal*

Get the title of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Title string representation

**Return type** *Literal*

**graph**

**id**

Identifier of the text

**Returns** Identifier of the text

**last**

Last CapitainsCtsPassage

**lastId**

Children passage

**Return type** `str`

**Returns** First children of the graph. Shortcut to `self.graph.children[0]`

**metadata**

Metadata information about the text

**Returns** Collection object with metadata about the text

**Return type** `Metadata`

**next**

Get Next TextualNode

**nextId**

Next Id

**parent**

Parent TextualNode

**parentId**

Parent Id

**prev**

Get Previous TextualNode

**prevId**

Previous Id (Sibling)

**static prevnext** (*resource*)

Parse a resource to get the prev and next urn

**Parameters** **resource** (*etree.\_Element*) – XML Resource

**Returns** Tuple representing previous and next urn

**Return type** (`str`, `str`)

**reffs**

Get all valid reffs for every part of the CtsText

**Return type** `MyCapytain.resources.texts.tei.XmlCtsCitation`

**retriever**

Retriever object used to query for more data

**Return type** `CitableTextServiceRetriever`

**set\_creator** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)

Set the DC Creator literal value

**Parameters**

- **value** – Value of the creator node
- **lang** – Language in which the value is

**set\_description** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)

Set the DC Description literal value

**Parameters**

- **value** – Value of the title node

- **lang** – Language in which the value is

**set\_metadata\_from\_collection** (*text\_metadata: MyCapytain.resources.prototypes.cts.inventory.CtsTextMetadata*)  
Set the object metadata using its collections recursively

**Parameters** **text\_metadata** (*CtsTextMetadata*) – Object representing the current text as a collection

**set\_subject** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)  
Set the DC Subject literal value

**Parameters**

- **value** – Value of the subject node
- **lang** – Language in which the value is

**set\_title** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)  
Set the DC Title literal value

**Parameters**

- **value** – Value of the title node
- **lang** – Language in which the value is

**siblingsId**  
Siblings Id

**text**  
String representation of the text

**Returns** String representation of the text

**urn**  
URN Identifier of the object

**class** MyCapytain.resources.texts.remote.cts.**CtsPassage** (*urn, resource, \*args, \*\*kwargs*)

Bases: MyCapytain.resources.texts.remote.cts.\_SharedMethod, MyCapytain.resources.prototypes.cts.text.PrototypeCtsPassage, MyCapytain.resources.texts.base.tei.TeiResource

CapitainsCtsPassage representing

**Parameters**

- **urn** –
- **resource** –
- **retriever** –
- **args** –
- **kwargs** –

**DEFAULT\_EXPORT** = 'text/plain'

**EXPORT\_TO** = ['python/lxml', 'text/xml', 'python/NestedDict', 'text/plain', 'text/xml:t

**PLAINTEXT\_STRING\_JOIN** = ' '

**asNode** () → rdflib.term.Identifier

**childIds**  
Identifiers of children

**Returns** Identifiers of children

**children**

Children TextualNode

**citation**

Citation system of the object

**Return type** *Citation*

**default\_exclude** = []

**depth**

Depth of the current object

**Returns** Int representation of the depth based on URN information

**Return type** *int*

**export** (*output: str = None, exclude: List[str] = None, \*\*kwargs*)

Export the collection item in the Mimetype required.

..note:: If current implementation does not have special mimetypes, reuses default\_export method

**Parameters**

- **output** (*str*) – Mimetype to export to (Uses MyCapytain.common.constants.Mimetypes)
- **exclude** (*[str]*) – Information to exclude. Specific to implementations

**Returns** Object using a different representation

**export\_capacities**

List Mimetypes that current object can export to

**first**

First TextualNode

**firstId**

Children passage

**Return type** *str*

**Returns** First children of the graph. Shortcut to self.graph.children[0]

**static firstUrn** (*resource*)

Parse a resource to get the first URN

**Parameters** **resource** (*etree.\_Element*) – XML Resource

**Returns** Tuple representing previous and next urn

**Return type** *str*

**getFirstUrn** (*reference=None*)

Get the first children URN for a given resource

**Parameters** **reference** (*CtsReference, str*) – CtsReference from which to find child  
(If None, find first reference)

**Returns** Children URN

**Return type** *URN*

**getLabel** ()

Retrieve metadata about the text

**Return type** Metadata

**Returns** Dictionary with label informations

**getPassagePlus** (*reference=None*)

Retrieve a passage and informations around it and store it in the object

**Parameters** **reference** (*CtsReference or List of text\_type*) – Reference of the passage

**Return type** *CtsPassage*

**Returns** Object representing the passage

**Raises** *TypeError* when reference is not a list or a Reference

**getPrevNextUrn** (*reference*)

Get the previous URN of a reference of the text

**Parameters** **reference** (*Union[CtsReference, str]*) – CtsReference from which to find siblings

**Returns** (Previous CapitainsCtsPassage CtsReference, Next CapitainsCtsPassage CtsReference)

**getReffs** (*level=1, subreference=None*)

Reference available at a given level

**Parameters**

- **level** (*Int*) – Depth required. If not set, should retrieve first encountered level (1 based)
- **subreference** (*CtsReference*) – Subreference (optional)

**Return type** [text\_type]

**Returns** List of levels

**getTextualNode** (*subreference=None*)

Retrieve a passage and store it in the object

**Parameters** **subreference** (*Union[CtsReference, URN, str, list]*) – CtsReference of the passage (Note : if given a list, this should be a list of string that compose the reference)

**Return type** *CtsPassage*

**Returns** Object representing the passage

**Raises** *TypeError* when reference is not a list or a CtsReference

**getValidReff** (*level=1, reference=None*)

Given a resource, CtsText will compute valid reffs

**Parameters**

- **level** (*Int*) – Depth required. If not set, should retrieve first encountered level (1 based)
- **reference** (*CtsReference*) – CapitainsCtsPassage reference

**Return type** list(str)

**Returns** List of levels

**get\_creator** (*lang: str = None*) → rdflib.term.Literal

Get the DC Creator literal value

**Parameters** **lang** – Language to retrieve

**Returns** Creator string representation

**Return type** Literal

**get\_cts\_metadata** (*key: str, lang: str = None*) → rdflib.term.Literal

Get easily a metadata from the CTS namespace

**Parameters**

- **key** – CTS property to retrieve
- **lang** – Language in which it should be

**Returns** Literal value of the CTS graph property

**get\_description** (*lang: str = None*) → rdflib.term.Literal

Get the description of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Description string representation

**Return type** Literal

**get\_subject** (*lang=None*) → rdflib.term.Literal

Get the subject of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Subject string representation

**Return type** Literal

**get\_title** (*lang: str = None*) → rdflib.term.Literal

Get the title of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Title string representation

**Return type** Literal

**graph**

**id**

Identifier of the text

**Returns** Identifier of the text

**last**

Last CapitainsCtsPassage

**lastId**

Children passage

**Return type** `str`

**Returns** First children of the graph. Shortcut to `self.graph.children[0]`

**metadata**

Metadata information about the text

**Returns** Collection object with metadata about the text

**Return type** Metadata

**next**

Get Next TextualNode

**nextId**

Shortcut for getting the following passage identifier

**Return type** *CtsReference*

**Returns** Following passage reference

**parent**

Parent TextualNode

**parentId**

Shortcut for getting the parent passage identifier

**Return type** *CtsReference*

**Returns** Following passage reference

**plaintext\_string\_join**

String used to join xml node's texts in export

**prev**

Get Previous TextualNode

**prevId**

Previous passage Identifier

**Return type** *CtsPassage*

**Returns** Previous passage at same level

**static prevnext** (*resource*)

Parse a resource to get the prev and next urn

**Parameters** **resource** (*etree.\_Element*) – XML Resource

**Returns** Tuple representing previous and next urn

**Return type** (*str, str*)

**reference****retriever**

Retriever object used to query for more data

**Return type** *CitableTextServiceRetriever*

**set\_creator** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)

Set the DC Creator literal value

**Parameters**

- **value** – Value of the creator node
- **lang** – Language in which the value is

**set\_description** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)

Set the DC Description literal value

**Parameters**

- **value** – Value of the title node
- **lang** – Language in which the value is

**set\_metadata\_from\_collection** (*text\_metadata: MyCapytain.resources.prototypes.cts.inventory.CtsTextMetadata*)

Set the object metadata using its collections recursively

**Parameters** `text_metadata` (*CtsTextMetadata*) – Object representing the current text as a collection

**set\_subject** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str]*, *lang: str = None*)  
Set the DC Subject literal value

**Parameters**

- **value** – Value of the subject node
- **lang** – Language in which the value is

**set\_title** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str]*, *lang: str = None*)  
Set the DC Title literal value

**Parameters**

- **value** – Value of the title node
- **lang** – Language in which the value is

**siblingsId**  
Shortcut for getting the previous and next passage identifier

**Return type** *CtsReference*

**Returns** Following passage reference

**text**  
String representation of the text  
**Returns** String representation of the text

**urn**  
URN Identifier of the object

**xml**  
XML Representation of the CapitainsCtsPassage

**Return type** *lxml.etree.\_Element*

**Returns** XML element representing the passage

## Collections

### Metadata

```
class MyCapytain.resources.prototypes.metadata.Collection (identifier="", *args,  
                                                    **kwargs)
```

Bases: *MyCapytain.common.base.Exportable*

Collection represents any resource's metadata. It has members and parents

**Variables**

- **properties** – Properties of the collection
- **parents** – Parent of the node from the direct parent to the highest ascendant
- **metadata** – Metadata

**DEFAULT\_EXPORT** = *None*

**EXPORT\_TO** = [*'application/ld+json'*, *'application/ld+json:DTS'*, *'application/rdf+xml'*]

**MODEL\_URI** = *rdflib.term.URIRef('https://w3id.org/dts/api#collection')*

**TYPE\_URI** = `rdflib.term.URIRef('https://w3id.org/dts/api#collection')`

**asNode** ()  
Node representation of the collection in the graph

**Return type** URIRef

**children**  
Dictionary of childrens {Identifier: Collection}

**Return type** dict

**descendants**  
Any descendant (no max level) of the collection's item

**Return type** [Collection]

**export** (*output=None, \*\*kwargs*)  
Export the collection item in the Mimetype required.

**Parameters** **output** (*str*) – Mimetype to export to (Uses MyCapytain.common.utils.Mimetypes)

**Returns** Object using a different representation

**classmethod export\_base\_dts** (*graph, obj, nsm*)  
Export the base DTS information in a simple reusable way

**Parameters**

- **graph** – Current graph where the information lie
- **obj** – Object for which we build info
- **nsm** – Namespace manager

**Returns** Dict

**export\_capacities**  
List Mimetypes that current object can export to

**get\_label** (*lang=None*)  
Return label for given lang or any default

**Parameters** **lang** – Language to request

**Returns** Label value

**Return type** Literal

**graph**  
RDFLib Graph space

**Return type** Graph

**id**

**members**  
Children of the collection's item

**Return type** [Collection]

**metadata**

**model**

**parent**  
Parent of current object

**Return type** Collection

**parents**

Iterator to find parents of current collection, from closest to furthest

**Return type** Generator[Collection]

**readable**

Readable property should return elements where the element can be queried for `getPassage / getReffs`

**readableDescendants**

List of element available which are readable

**Return type** [Collection]

**set\_label** (*label, lang*)

Add the label of the collection in given lang

**Parameters**

- **label** – Label Value
- **lang** – Language code

**size**

**type**

**version**

```
class MyCapytain.resources.prototypes.metadata.ResourceCollection (identifier=",
                                                                    *args,
                                                                    **kwargs)
```

Bases: `MyCapytain.resources.prototypes.metadata.Collection`

**DEFAULT\_EXPORT** = None

**EXPORT\_TO** = ['application/ld+json', 'application/ld+json:DTS', 'application/rdf+xml']

**MODEL\_URI** = `rdflib.term.URIRef('https://w3id.org/dts/api#collection')`

**TYPE\_URI** = `rdflib.term.URIRef('https://w3id.org/dts/api#collection')`

**asNode** ()

Node representation of the collection in the graph

**Return type** URIRef

**children**

Dictionary of childrens {Identifier: Collection}

**Return type** dict

**descendants**

Any descendant (no max level) of the collection's item

**Return type** [Collection]

**export** (*output=None, \*\*kwargs*)

Export the collection item in the Mimetype required.

**Parameters** **output** (*str*) – Mimetype to export to (Uses `MyCapytain.common.utils.Mimetypes`)

**Returns** Object using a different representation

**classmethod export\_base\_dts** (*graph, obj, nsm*)

Export the base DTS information in a simple reusable way

**Parameters**

- **graph** – Current graph where the information lie
- **obj** – Object for which we build info
- **nsm** – Namespace manager

**Returns** Dict**export\_capacities**

List Mimetypes that current object can export to

**get\_creator** (*lang=None*)

Get the DC Creator literal value

**Parameters** **lang** – Language to retrieve**Returns** Creator string representation**Return type** Literal**get\_description** (*lang=None*)

Get the description of the object

**Parameters** **lang** – Lang to retrieve**Returns** Description string representation**Return type** Literal**get\_label** (*lang=None*)

Return label for given lang or any default

**Parameters** **lang** – Language to request**Returns** Label value**Return type** Literal**get\_subject** (*lang=None*)

Get the subject of the object

**Parameters** **lang** – Lang to retrieve**Returns** Subject string representation**Return type** Literal**get\_title** (*lang=None*)

Get the title of the object

**Parameters** **lang** – Lang to retrieve**Returns** Title string representation**Return type** Literal**graph**

RDFLib Graph space

**Return type** Graph**id****lang**

Languages this text is in

**Returns** List of available languages

**members**

Children of the collection's item

**Return type** [Collection]

**metadata**

**model**

**parent**

Parent of current object

**Return type** Collection

**parents**

Iterator to find parents of current collection, from closest to furthest

**Return type** Generator[Collection]

**readable**

Readable property should return elements where the element can be queried for `getPassage / getReffs`

**readableDescendants**

List of element available which are readable

**Return type** [Collection]

**set\_label** (*label, lang*)

Add the label of the collection in given lang

**Parameters**

- **label** – Label Value
- **lang** – Language code

**size**

**type**

**version**

## CTS inventory

```
class MyCapytain.resources.collections.cts.XmlCtsCitation (name=None,  
xpath=None,  
scope=None,  
refsDecl=None,  
child=None)
```

Bases: `MyCapytain.common.reference._capitains_cts.Citation`

XmlCtsCitation XML implementation for CtsTextInventoryMetadata

```
DEFAULT_EXPORT = 'text/xml:CTS'
```

```
EXPORT_TO = ['text/xml:CTS', 'text/xml:tei']
```

**add\_child** (*child*)

Adds a child to the CitationSet

**Parameters** **child** – Child citation to add

**Returns**

**attribute**

Attribute that serves as a reference getter

**child**

Child of a citation

**Type** XmlCtsCitation or None

**Example** XmlCtsCitation.name==poem would have a child XmlCtsCitation.name==line

**children**

Children of a citation

**Return type** [BaseCitation]

**depth**

Depth of the citation scheme

**Return type** int

**Returns** Depth of the citation scheme

**escape = re.compile('(")')**

**export** (*output=None, \*\*kwargs*)

Export the collection item in the Mime type required.

**Parameters** **output** (*str*) – Mime type to export to (Uses MyCapytain.common.utils.Mimetypes)

**Returns** Object using a different representation

**export\_capacities**

List Mimetypes that current object can export to

**fill** (*passage=None, xpath=None*)

Fill the xpath with given informations

**Parameters**

- **passage** (*CtsReference or list or None. Can be list of None and not None*) – CapitainsCtsPassage reference
- **xpath** (*Boolean*) – If set to True, will return the replaced self.xpath value and not the whole self.refsDecl

**Return type** basestring

**Returns** Xpath to find the passage

```

citation = XmlCtsCitation(name="line", scope="/TEI/text/body/div/div[@n=?]",
↳xpath="//l[@n=?]")
print(citation.fill(["1", None]))
# /TEI/text/body/div/div[@n='1']//l[@n]
print(citation.fill(None))
# /TEI/text/body/div/div[@n]//l[@n]
print(citation.fill(CtsReference("1.1")))
# /TEI/text/body/div/div[@n='1']//l[@n='1']
print(citation.fill("1", xpath=True))
# //l[@n='1']

```

**classmethod ingest** (*resource, element=None, xpath='ti:citation'*)

Ingest xml to create a citation

**Parameters**

- **resource** – XML on which to do xpath
- **element** – Element where the citation should be stored
- **xpath** – XPath to use to retrieve citation

**Returns** XmlCtsCitation

**is\_empty** () → bool

Check if the citation has not been set

**Returns** True if nothing was setup

**Return type** bool

**is\_root** () → str

**Returns** If the current object is the root of the citation set, True

**Return type** bool

**is\_set** () → bool

Check if the citation has been set

**Returns** True if set up, False if not

**Return type** bool

**match** (*passageId*)

Given a passageId matches a citation level

**Parameters** **passageId** – A passage to match

**Returns**

**name**

Type of the citation represented

**Return type** str

**Example** Book, Chapter, Textpart, Section, Poem...

**refsDecl**

ResfDecl expression of the citation scheme

**Return type** str

**Example** /tei:TEI/tei:text/tei:body/tei:div//tei:l[@n='\$1']

**root**

Returns the root of the citation set

**Returns** Root of the Citation set

**Return type** *BaseCitationSet*

**scope**

CtsTextInventoryMetadata scope property of a citation (ie. identifier of all element but the last of the citation)

**Type** basestring

**Example** /tei:TEI/tei:text/tei:body/tei:div

**xpath**

CtsTextInventoryMetadata xpath property of a citation (ie. identifier of the last element of the citation)

**Type** basestring

**Example** //tei:l[@n="???"]

```
class MyCapytain.resources.collections.cts.XmlCtsWorkMetadata (urn=None, parent=None)
    Bases: MyCapytain.resources.prototypes.cts.inventory.CtsWorkMetadata
    Represents a CTS Textgroup in XML

CLASS_COMMENTARY
    alias of XmlCtsCommentaryMetadata

CLASS_EDITION
    alias of XmlCtsEditionMetadata

CLASS_TRANSLATION
    alias of XmlCtsTranslationMetadata

CTS_LINKS = []

CTS_PROPERTIES = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/title')]

DC_TITLE_KEY = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/title')

DEFAULT_EXPORT = 'python/lxml'

EXPORT_TO = ['text/xml:CTS', 'text/xml:CTS_CapiTainS']

MODEL_URI = rdflib.term.URIRef('https://w3id.org/dts/api#collection')

TYPE_URI = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/work')

asNode ()
    Node representation of the collection in the graph

    Return type URIRef

children
    Dictionary of childrens {Identifier: Collection}

    Return type dict

descendants
    Any descendant (no max level) of the collection's item

    Return type [Collection]

export (output=None, **kwargs)
    Export the collection item in the Mimetype required.

    Parameters output (str) - Mimetype to export to (Uses MyCapytain.common.utils.Mimetypes)

    Returns Object using a different representation

classmethod export_base_dts (graph, obj, nsm)
    Export the base DTS information in a simple reusable way

    Parameters

- graph - Current graph where the information lie
- obj - Object for which we build info
- nsm - Namespace manager

Returns Dict
```

**export\_capacities**

List Mimetypes that current object can export to

**get\_cts\_property** (*prop*, *lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to get (Without namespace)
- **lang** – Language to get for given value

**Returns** Value or default if lang is set, else whole set of values

**Return type** dict or Literal

**get\_label** (*lang=None*)

Return label for given lang or any default

**Parameters** **lang** – Language to request

**Returns** Label value

**Return type** Literal

**get\_link** (*prop*)

Get given link in CTS Namespace

**Parameters** **prop** – Property to get (Without namespace)

**Returns** whole set of values

**Return type** list

**get\_translation\_in** (*key=None*)

Find a translation with given language

**Parameters** **key** (*text\_type*) – Language to find

**Return type** [CtsTextMetadata]

**Returns** List of available translations

**graph**

RDFLib Graph space

**Return type** Graph

**id**

**lang**

Languages this text is in

**Returns** List of available languages

**members**

Children of the collection's item

**Return type** [Collection]

**metadata**

**model**

**parent**

Parent of current object

**Return type** Collection

**parents**

Iterator to find parents of current collection, from closest to furthest

**Return type** Generator[Collection]

**classmethod parse** (*resource*, *parent=None*, *\_with\_children=False*)

Parse a resource

**Parameters**

- **resource** – Element representing a work
- **parent** (*XmlCtsTextgroupMetadata*) – Parent of the object

**readable**

Readable property should return elements where the element can be queried for `getPassage / getReffs`

**readableDescendants**

List of element available which are readable

**Return type** [Collection]

**set\_cts\_property** (*prop*, *value*, *lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property
- **lang** – Language to set for given value

**set\_label** (*label*, *lang*)

Add the label of the collection in given lang

**Parameters**

- **label** – Label Value
- **lang** – Language code

**set\_link** (*prop*, *value*)

Set given link in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property

**size****texts**

Texts

**Returns** Dictionary of texts

**Return type** defaultdict(PrototypeTexts)

**type****update** (*other*)

Merge two XmlCtsWorkMetadata Objects.

- Original (left Object) keeps his parent.
- Added document overwrite text if it already exists

**Parameters** *other* (*CtsWorkMetadata*) – XmlCtsWorkMetadata object

**Returns** XmlCtsWorkMetadata Object

**Rtype** XmlCtsWorkMetadata

**urn**

**version**

**class** MyCapytain.resources.collections.cts.XmlCtsCommentaryMetadata (\*args, \*\*kwargs)

Bases: MyCapytain.resources.prototypes.cts.inventory.CtsCommentaryMetadata, MyCapytain.resources.collections.cts.XmlCtsTextMetadata

Create a commentary subtyped PrototypeText object

**CLASS\_CITATION**

alias of XmlCtsCitation

**CTS\_LINKS** = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/about')]

**CTS\_PROPERTIES** = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/label'), rdflib

**DC\_TITLE\_KEY** = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/label')

**DEFAULT\_EXPORT** = 'python/lxml'

**EXPORT\_TO** = ['text/xml:CTS', 'text/xml:CTS\_CapiTainS']

**MODEL\_URI** = rdflib.term.URIRef('https://w3id.org/dts/api#resource')

**SUBTYPE** = 'commentary'

**TYPE\_URI** = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/commentary')

**asNode()**

Node representation of the collection in the graph

**Return type** URIRef

**children**

Dictionary of childrens {Identifier: Collection}

**Return type** dict

**descendants**

Descendants of the collection's item

**Warning:** CapitainsCtsText has no Descendants

**Return type** list

**editions()**

Get all editions of the texts

**Returns** List of editions

**Return type** [CtsTextMetadata]

**export** (*output=None, \*\*kwargs*)

Export the collection item in the Mimetype required.

**Parameters** **output** (*str*) – Mimetype to export to (Uses MyCapytain.common.utils.Mimetypes)

**Returns** Object using a different representation

**classmethod** `export_base_dts` (*graph, obj, nsm*)  
Export the base DTS information in a simple reusable way

**Parameters**

- **graph** – Current graph where the information lie
- **obj** – Object for which we build info
- **nsm** – Namespace manager

**Returns** Dict

**export\_capacities**  
List Mimetypes that current object can export to

**get\_creator** (*lang=None*)  
Get the DC Creator literal value

**Parameters** **lang** – Language to retrieve

**Returns** Creator string representation

**Return type** Literal

**get\_cts\_property** (*prop, lang=None*)  
Set given property in CTS Namespace

**Parameters**

- **prop** – Property to get (Without namespace)
- **lang** – Language to get for given value

**Returns** Value or default if lang is set, else whole set of values

**Return type** dict or Literal

**get\_description** (*lang=None*)  
Get the DC description of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Description string representation

**Return type** Literal

**get\_label** (*lang=None*)  
Return label for given lang or any default

**Parameters** **lang** – Language to request

**Returns** Label value

**Return type** Literal

**get\_link** (*prop*)  
Get given link in CTS Namespace

**Parameters** **prop** – Property to get (Without namespace)

**Returns** whole set of values

**Return type** list

**get\_subject** (*lang=None*)  
Get the DC subject of the object

**Parameters** `lang` – Lang to retrieve

**Returns** Subject string representation

**Return type** Literal

**get\_title** (*lang=None*)

Get the DC Title of the object

**Parameters** `lang` – Lang to retrieve

**Returns** Title string representation

**Return type** Literal

**graph**

RDFLib Graph space

**Return type** Graph

**id**

**lang**

Languages this text is in

**Returns** List of available languages

**members**

Children of the collection's item

**Warning:** CapitainsCtsText has no children

**Return type** `list`

**metadata**

**model**

**parent**

Parent of current object

**Return type** Collection

**parents**

Iterator to find parents of current collection, from closest to furthest

**Return type** Generator[Collection]

**classmethod parse** (*resource, parent=None*)

**classmethod parse\_metadata** (*obj, xml*)

Parse a resource to feed the object

**Parameters**

- **obj** (*XmlCtsTextMetadata*) – Obj to set metadata of
- **xml** (*lxml.etree.\_Element*) – An xml representation object

**path**

**readable**

Readable property should return elements where the element can be queried for `getPassage / getReffs`

**readableDescendants**

List of element available which are readable

**Return type** [Collection]

**set\_cts\_property** (*prop, value, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property
- **lang** – Language to set for given value

**set\_label** (*label, lang*)

Add the label of the collection in given lang

**Parameters**

- **label** – Label Value
- **lang** – Language code

**set\_link** (*prop, value*)

Set given link in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property

**size****subtype**

Subtype of the object

**Returns** string representation of subtype

**translations** (*key=None*)

Get translations in given language

**Parameters** **key** – Language ISO Code to filter on

**Returns**

**type****urn****version**

```
class MyCapytain.resources.collections.cts.XmlCtsTranslationMetadata (*args,
                                                                    **kwargs)
```

```
Bases: MyCapytain.resources.prototypes.cts.inventory.CtsTranslationMetadata,
MyCapytain.resources.collections.cts.XmlCtsTextMetadata
```

Create a translation subtyped CtsTextMetadata object

**CLASS CITATION**

alias of XmlCtsCitation

```
CTS_LINKS = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/about')]
```

```
CTS_PROPERTIES = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/label'), rdflib
```

```
DC_TITLE_KEY = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/label')
```

```

DEFAULT_EXPORT = 'python/lxml'
EXPORT_TO = ['text/xml:CTS', 'text/xml:CTS_CapiTainS']
MODEL_URI = rdflib.term.URIRef('https://w3id.org/dts/api#resource')
SUBTYPE = 'translation'
TYPE_URI = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/translation')

```

**asNode()**

Node representation of the collection in the graph

**Return type** `URIRef`

**children**

Dictionary of childrens {Identifier: Collection}

**Return type** `dict`

**descendants**

Descendants of the collection's item

**Warning:** CapitainsCtsText has no Descendants

**Return type** `list`

**editions()**

Get all editions of the texts

**Returns** List of editions

**Return type** `[CtsTextMetadata]`

**export** (*output=None, \*\*kwargs*)

Export the collection item in the Mime type required.

**Parameters** **output** (*str*) – Mime type to export to (Uses `MyCapytain.common.utils.Mimetypes`)

**Returns** Object using a different representation

**classmethod export\_base\_dts** (*graph, obj, nsm*)

Export the base DTS information in a simple reusable way

**Parameters**

- **graph** – Current graph where the information lie
- **obj** – Object for which we build info
- **nsm** – Namespace manager

**Returns** `Dict`

**export\_capacities**

List Mime types that current object can export to

**get\_creator** (*lang=None*)

Get the DC Creator literal value

**Parameters** **lang** – Language to retrieve

**Returns** Creator string representation

**Return type** Literal

**get\_cts\_property** (*prop*, *lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to get (Without namespace)
- **lang** – Language to get for given value

**Returns** Value or default if lang is set, else whole set of values

**Return type** dict or Literal

**get\_description** (*lang=None*)

Get the DC description of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Description string representation

**Return type** Literal

**get\_label** (*lang=None*)

Return label for given lang or any default

**Parameters** **lang** – Language to request

**Returns** Label value

**Return type** Literal

**get\_link** (*prop*)

Get given link in CTS Namespace

**Parameters** **prop** – Property to get (Without namespace)

**Returns** whole set of values

**Return type** list

**get\_subject** (*lang=None*)

Get the DC subject of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Subject string representation

**Return type** Literal

**get\_title** (*lang=None*)

Get the DC Title of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Title string representation

**Return type** Literal

**graph**

RDFLib Graph space

**Return type** Graph

**id**

**lang**

Languages this text is in

**Returns** List of available languages

**members**

Children of the collection's item

**Warning:** CapitainsCtsText has no children

**Return type** `list`

**metadata**

**model**

**parent**

Parent of current object

**Return type** `Collection`

**parents**

Iterator to find parents of current collection, from closest to furthest

**Return type** `Generator[Collection]`

**classmethod** `parse` (*resource*, *parent=None*)

**classmethod** `parse_metadata` (*obj*, *xml*)

Parse a resource to feed the object

**Parameters**

- **obj** (`XmlCtsTextMetadata`) – Obj to set metadata of
- **xml** (`lxml.etree._Element`) – An xml representation object

**path**

**readable**

Readable property should return elements where the element can be queried for `getPassage` / `getReffs`

**readableDescendants**

List of element available which are readable

**Return type** `[Collection]`

**set\_cts\_property** (*prop*, *value*, *lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property
- **lang** – Language to set for given value

**set\_label** (*label*, *lang*)

Add the label of the collection in given lang

**Parameters**

- **label** – Label Value
- **lang** – Language code

**set\_link** (*prop, value*)

Set given link in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property

**size**

**subtype**

Subtype of the object

**Returns** string representation of subtype

**translations** (*key=None*)

Get translations in given language

**Parameters** **key** – Language ISO Code to filter on

**Returns**

**type**

**urn**

**version**

```
class MyCapytain.resources.collections.cts.XmlCtsEditionMetadata (*args,
                                                                **kwargs)
```

Bases: MyCapytain.resources.prototypes.cts.inventory.CtsEditionMetadata,  
MyCapytain.resources.collections.cts.XmlCtsTextMetadata

Create an edition subtyped CtsTextMetadata object

**CLASS\_CITATION**

alias of XmlCtsCitation

**CTS\_LINKS** = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/about')]

**CTS\_PROPERTIES** = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/label'), rdflib

**DC\_TITLE\_KEY** = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/label')

**DEFAULT\_EXPORT** = 'python/lxml'

**EXPORT\_TO** = ['text/xml:CTS', 'text/xml:CTS\_CapiTainS']

**MODEL\_URI** = rdflib.term.URIRef('https://w3id.org/dts/api#resource')

**SUBTYPE** = 'edition'

**TYPE\_URI** = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/edition')

**asNode** ()

Node representation of the collection in the graph

**Return type** URIRef

**children**

Dictionary of childrens {Identifier: Collection}

**Return type** dict

**descendants**

Descendants of the collection's item

**Warning:** CapitainsCtsText has no Descendants

**Return type** `list`

**editions** ()

Get all editions of the texts

**Returns** List of editions

**Return type** `[CtsTextMetadata]`

**export** (*output=None, \*\*kwargs*)

Export the collection item in the Mimetype required.

**Parameters** **output** (*str*) – Mimetype to export to (Uses `MyCapytain.common.utils.Mimetypes`)

**Returns** Object using a different representation

**classmethod export\_base\_dts** (*graph, obj, nsm*)

Export the base DTS information in a simple reusable way

**Parameters**

- **graph** – Current graph where the information lie
- **obj** – Object for which we build info
- **nsm** – Namespace manager

**Returns** Dict

**export\_capacities**

List Mimetypes that current object can export to

**get\_creator** (*lang=None*)

Get the DC Creator literal value

**Parameters** **lang** – Language to retrieve

**Returns** Creator string representation

**Return type** Literal

**get\_cts\_property** (*prop, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to get (Without namespace)
- **lang** – Language to get for given value

**Returns** Value or default if lang is set, else whole set of values

**Return type** `dict` or Literal

**get\_description** (*lang=None*)

Get the DC description of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Description string representation

**Return type** Literal

**get\_label** (*lang=None*)

Return label for given lang or any default

**Parameters** **lang** – Language to request

**Returns** Label value

**Return type** Literal

**get\_link** (*prop*)

Get given link in CTS Namespace

**Parameters** **prop** – Property to get (Without namespace)

**Returns** whole set of values

**Return type** list

**get\_subject** (*lang=None*)

Get the DC subject of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Subject string representation

**Return type** Literal

**get\_title** (*lang=None*)

Get the DC Title of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Title string representation

**Return type** Literal

**graph**

RDFLib Graph space

**Return type** Graph

**id**

**lang**

Languages this text is in

**Returns** List of available languages

**members**

Children of the collection's item

**Warning:** CapitainsCtsText has no children

**Return type** list

**metadata**

**model**

**parent**

Parent of current object

**Return type** Collection

**parents**

Iterator to find parents of current collection, from closest to furthest

**Return type** Generator[Collection]

**classmethod parse** (*resource*, *parent=None*)

**classmethod parse\_metadata** (*obj*, *xml*)

Parse a resource to feed the object

**Parameters**

- **obj** (*XmlCtsTextMetadata*) – Obj to set metadata of
- **xml** (*lxml.etree.\_Element*) – An xml representation object

**path**

**readable**

Readable property should return elements where the element can be queried for `getPassage / getReffs`

**readableDescendants**

List of element available which are readable

**Return type** [Collection]

**set\_cts\_property** (*prop*, *value*, *lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property
- **lang** – Language to set for given value

**set\_label** (*label*, *lang*)

Add the label of the collection in given lang

**Parameters**

- **label** – Label Value
- **lang** – Language code

**set\_link** (*prop*, *value*)

Set given link in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property

**size**

**subtype**

Subtype of the object

**Returns** string representation of subtype

**translations** (*key=None*)

Get translations in given language

**Parameters** **key** – Language ISO Code to filter on

**Returns**

**type**

**urn**

**version**

```
class MyCapytain.resources.collections.cts.XmlCtsTextgroupMetadata (urn=",
                                                                    par-
                                                                    ent=None)
Bases: MyCapytain.resources.prototypes.cts.inventory.CtsTextgroupMetadata
```

Represents a CTS Textgroup in XML

**CLASS\_WORK**

alias of XmlCtsWorkMetadata

**CTS\_LINKS** = []

**CTS\_PROPERTIES** = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/groupname')]

**DC\_TITLE\_KEY** = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/groupname')

**DEFAULT\_EXPORT** = 'python/lxml'

**EXPORT\_TO** = ['text/xml:CTS', 'text/xml:CTS\_CapiTainS']

**MODEL\_URI** = rdflib.term.URIRef('https://w3id.org/dts/api#collection')

**TYPE\_URI** = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/textgroup')

**asNode()**

Node representation of the collection in the graph

**Return type** URIRef

**children**

Dictionary of childrens {Identifier: Collection}

**Return type** dict

**descendants**

Any descendant (no max level) of the collection's item

**Return type** [Collection]

**export** (*output=None, \*\*kwargs*)

Export the collection item in the Mimetype required.

**Parameters** **output** (*str*) – Mimetype to export to (Uses MyCapytain.common.utils.Mimetypes)

**Returns** Object using a different representation

**classmethod** **export\_base\_dts** (*graph, obj, nsm*)

Export the base DTS information in a simple reusable way

**Parameters**

- **graph** – Current graph where the information lie
- **obj** – Object for which we build info
- **nsm** – Namespace manager

**Returns** Dict

**export\_capacities**

List Mimetypes that current object can export to

**get\_cts\_property** (*prop*, *lang=None*)  
Set given property in CTS Namespace

**Parameters**

- **prop** – Property to get (Without namespace)
- **lang** – Language to get for given value

**Returns** Value or default if lang is set, else whole set of values

**Return type** dict or Literal

**get\_label** (*lang=None*)  
Return label for given lang or any default

**Parameters** **lang** – Language to request

**Returns** Label value

**Return type** Literal

**get\_link** (*prop*)  
Get given link in CTS Namespace

**Parameters** **prop** – Property to get (Without namespace)

**Returns** whole set of values

**Return type** list

**graph**  
RDFLib Graph space

**Return type** Graph

**id**

**members**  
Children of the collection's item

**Return type** [Collection]

**metadata**

**model**

**parent**  
Parent of current object

**Return type** Collection

**parents**  
Iterator to find parents of current collection, from closest to furthest

**Return type** Generator[Collection]

**classmethod parse** (*resource*, *parent=None*)  
Parse a textgroup resource

**Parameters**

- **resource** – Element representing the textgroup
- **parent** – Parent of the textgroup

**readable**  
Readable property should return elements where the element can be queried for getPassage / getReffs

**readableDescendants**

List of element available which are readable

**Return type** [Collection]

**set\_cts\_property** (*prop, value, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property
- **lang** – Language to set for given value

**set\_label** (*label, lang*)

Add the label of the collection in given lang

**Parameters**

- **label** – Label Value
- **lang** – Language code

**set\_link** (*prop, value*)

Set given link in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property

**size****type****update** (*other*)

Merge two Textgroup Objects.

- Original (left Object) keeps his parent.
- Added document merges with work if it already exists

**Parameters** **other** (*CtsTextgroupMetadata*) – Textgroup object

**Returns** Textgroup Object

**Return type** CtsTextgroupMetadata

**urn****version****works**

Works

**Returns** Dictionary of works

**Return type** defaultdict(PrototypeWorks)

```
class MyCapytain.resources.collections.cts.XmlCtsTextInventoryMetadata (name='defaultInventory',
                                                                    par-
                                                                    ent=None)
```

Bases:  
CtsTextInventoryMetadata

MyCapytain.resources.prototypes.cts.inventory.

Represents a CTS Inventory file

**CLASS\_TEXTGROUP**

alias of XmlCtsTextgroupMetadata

**CTS\_LINKS** = []

**CTS\_PROPERTIES** = []

**DC\_TITLE\_KEY** = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/name')

**DEFAULT\_EXPORT** = 'python/1xml'

**EXPORT\_TO** = ['text/xml:CTS', 'text/xml:CTS\_CapiTainS']

**MODEL\_URI** = rdflib.term.URIRef('https://w3id.org/dts/api#collection')

**TYPE\_URI** = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/TextInventory')

**asNode()**

Node representation of the collection in the graph

**Return type** URIRef

**children**

Dictionary of childrens {Identifier: Collection}

**Return type** dict

**descendants**

Any descendant (no max level) of the collection's item

**Return type** [Collection]

**export** (*output=None, \*\*kwargs*)

Export the collection item in the Mimetype required.

**Parameters** **output** (*str*) – Mimetype to export to (Uses MyCapytain.common.utils.Mimetypes)

**Returns** Object using a different representation

**classmethod export\_base\_dts** (*graph, obj, nsm*)

Export the base DTS information in a simple reusable way

**Parameters**

- **graph** – Current graph where the information lie
- **obj** – Object for which we build info
- **nsm** – Namespace manager

**Returns** Dict

**export\_capacities**

List Mimetypes that current object can export to

**get\_cts\_property** (*prop, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to get (Without namespace)
- **lang** – Language to get for given value

**Returns** Value or default if lang is set, else whole set of values

**Return type** dict or Literal

**get\_label** (*lang=None*)

Return label for given lang or any default

**Parameters** **lang** – Language to request

**Returns** Label value

**Return type** Literal

**get\_link** (*prop*)

Get given link in CTS Namespace

**Parameters** **prop** – Property to get (Without namespace)

**Returns** whole set of values

**Return type** list

**graph**

RDFLib Graph space

**Return type** Graph

**id**

**members**

Children of the collection's item

**Return type** [Collection]

**metadata**

**model**

**parent**

Parent of current object

**Return type** Collection

**parents**

Iterator to find parents of current collection, from closest to furthest

**Return type** Generator[Collection]

**classmethod parse** (*resource*)

Parse a resource

**Parameters** **resource** – Element representing the text inventory

**readable**

Readable property should return elements where the element can be queried for getPassage / getReffs

**readableDescendants**

List of element available which are readable

**Return type** [Collection]

**set\_cts\_property** (*prop, value, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property

- **lang** – Language to set for given value

**set\_label** (*label, lang*)

Add the label of the collection in given lang

**Parameters**

- **label** – Label Value
- **lang** – Language code

**set\_link** (*prop, value*)

Set given link in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property

**size**

**textgroups**

Textgroups

**Returns** Dictionary of textgroups

**Return type** defaultdict(CtsTextgroupMetadata)

**type**

**urn**

**version**

**class** MyCapytain.resources.collections.cts.**XmlCtsTextMetadata** (*\*args, \*\*kwargs*)

Bases: MyCapytain.resources.prototypes.cts.inventory.CtsTextMetadata

Represents a CTS CtsTextMetadata

**CLASS\_CITATION**

alias of XmlCtsCitation

**CTS\_LINKS** = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/about')]

**CTS\_PROPERTIES** = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/label'), rdflib

**DC\_TITLE\_KEY** = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/label')

**DEFAULT\_EXPORT** = 'python/lxml'

**EXPORT\_TO** = ['text/xml:CTS', 'text/xml:CTS\_CapiTainS']

**MODEL\_URI** = rdflib.term.URIRef('https://w3id.org/dts/api#resource')

**SUBTYPE** = 'unknown'

**TYPE\_URI** = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/text')

**asNode** ()

Node representation of the collection in the graph

**Return type** URIRef

**children**

Dictionary of childrens {Identifier: Collection}

**Return type** dict

**descendants**

Descendants of the collection's item

**Warning:** CapitainsCtsText has no Descendants

**Return type** `list`

**editions** ()

Get all editions of the texts

**Returns** List of editions

**Return type** `[CtsTextMetadata]`

**export** (*output=None, \*\*kwargs*)

Export the collection item in the Mimetype required.

**Parameters** **output** (*str*) – Mimetype to export to (Uses `MyCapytain.common.utils.Mimetypes`)

**Returns** Object using a different representation

**classmethod export\_base\_dts** (*graph, obj, nsm*)

Export the base DTS information in a simple reusable way

**Parameters**

- **graph** – Current graph where the information lie
- **obj** – Object for which we build info
- **nsm** – Namespace manager

**Returns** Dict

**export\_capacities**

List Mimetypes that current object can export to

**get\_creator** (*lang=None*)

Get the DC Creator literal value

**Parameters** **lang** – Language to retrieve

**Returns** Creator string representation

**Return type** Literal

**get\_cts\_property** (*prop, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to get (Without namespace)
- **lang** – Language to get for given value

**Returns** Value or default if lang is set, else whole set of values

**Return type** `dict` or Literal

**get\_description** (*lang=None*)

Get the DC description of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Description string representation

**Return type** Literal

**get\_label** (*lang=None*)

Return label for given lang or any default

**Parameters** **lang** – Language to request

**Returns** Label value

**Return type** Literal

**get\_link** (*prop*)

Get given link in CTS Namespace

**Parameters** **prop** – Property to get (Without namespace)

**Returns** whole set of values

**Return type** list

**get\_subject** (*lang=None*)

Get the DC subject of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Subject string representation

**Return type** Literal

**get\_title** (*lang=None*)

Get the DC Title of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Title string representation

**Return type** Literal

**graph**

RDFLib Graph space

**Return type** Graph

**id**

**lang**

Languages this text is in

**Returns** List of available languages

**members**

Children of the collection's item

**Warning:** CapitainsCtsText has no children

**Return type** list

**metadata**

**model**

**parent**

Parent of current object

**Return type** Collection

**parents**

Iterator to find parents of current collection, from closest to furthest

**Return type** Generator[Collection]

**classmethod parse\_metadata** (*obj, xml*)

Parse a resource to feed the object

**Parameters**

- **obj** (*XmlCtsTextMetadata*) – Obj to set metadata of
- **xml** (*lxml.etree.\_Element*) – An xml representation object

**path**

**readable**

Readable property should return elements where the element can be queried for `getPassage / getReffs`

**readableDescendants**

List of element available which are readable

**Return type** [Collection]

**set\_cts\_property** (*prop, value, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property
- **lang** – Language to set for given value

**set\_label** (*label, lang*)

Add the label of the collection in given lang

**Parameters**

- **label** – Label Value
- **lang** – Language code

**set\_link** (*prop, value*)

Set given link in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property

**size**

**subtype**

Subtype of the object

**Returns** string representation of subtype

**translations** (*key=None*)

Get translations in given language

**Parameters** **key** – Language ISO Code to filter on

**Returns**

`type`  
`urn`  
`version`

## CTS Inventory Prototypes

**class** `MyCapytain.resources.prototypes.cts.inventory.PrototypeCtsCollection` (*identifier=""*)  
 Bases: `MyCapytain.resources.prototypes.metadata.Collection`

Resource represents any resource from the inventory

**Parameters** `identifier` (*str, URN*) – Identifier representing the CtsTextInventoryMetadata

**Variables** `CTS_MODEL` – String Representation of the type of collection

`CTS_LINKS` = []

`CTS_PROPERTIES` = []

`DC_TITLE_KEY` = None

`DEFAULT_EXPORT` = 'python/lxml'

`EXPORT_TO` = ['python/lxml']

`MODEL_URI` = `rdflib.term.URIRef('https://w3id.org/dts/api#collection')`

`TYPE_URI` = `rdflib.term.URIRef('https://w3id.org/dts/api#collection')`

`asNode` ()

Node representation of the collection in the graph

**Return type** `URIRef`

`children`

Dictionary of childrens {Identifier: Collection}

**Return type** `dict`

`descendants`

Any descendant (no max level) of the collection's item

**Return type** [Collection]

`export` (*output=None, \*\*kwargs*)

Export the collection item in the Mimetype required.

**Parameters** `output` (*str*) – Mimetype to export to (Uses `MyCapytain.common.utils.Mimetypes`)

**Returns** Object using a different representation

**classmethod** `export_base_dts` (*graph, obj, nsm*)

Export the base DTS information in a simple reusable way

**Parameters**

- `graph` – Current graph where the information lie
- `obj` – Object for which we build info
- `nsm` – Namespace manager

**Returns** Dict

**export\_capacities**

List Mimetypes that current object can export to

**get\_cts\_property** (*prop*, *lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to get (Without namespace)
- **lang** – Language to get for given value

**Returns** Value or default if lang is set, else whole set of values

**Return type** dict or Literal

**get\_label** (*lang=None*)

Return label for given lang or any default

**Parameters** **lang** – Language to request

**Returns** Label value

**Return type** Literal

**get\_link** (*prop*)

Get given link in CTS Namespace

**Parameters** **prop** – Property to get (Without namespace)

**Returns** whole set of values

**Return type** list

**graph**

RDFLib Graph space

**Return type** Graph

**id****members**

Children of the collection's item

**Return type** [Collection]

**metadata****model****parent**

Parent of current object

**Return type** Collection

**parents**

Iterator to find parents of current collection, from closest to furthest

**Return type** Generator[Collection]

**readable**

Readable property should return elements where the element can be queried for getPassage / getReffs

**readableDescendants**

List of element available which are readable

**Return type** [Collection]

**set\_cts\_property** (*prop, value, lang=None*)  
 Set given property in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property
- **lang** – Language to set for given value

**set\_label** (*label, lang*)  
 Add the label of the collection in given lang

**Parameters**

- **label** – Label Value
- **lang** – Language code

**set\_link** (*prop, value*)  
 Set given link in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property

**size**

**type**

**urn**

**version**

**class** MyCapytain.resources.prototypes.cts.inventory.CtsTextInventoryCollection (*identifier='default'*)  
 Bases: MyCapytain.resources.prototypes.cts.inventory.PrototypeCtsCollection

Initiate a CtsTextInventoryMetadata resource

**Parameters**

- **resource** (*Any*) – Resource representing the CtsTextInventoryMetadata
- **name** (*str*) – Identifier of the CtsTextInventoryMetadata

**CTS\_LINKS** = []

**CTS\_PROPERTIES** = []

**DC\_TITLE\_KEY** = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/name')

**DEFAULT\_EXPORT** = 'python/lxml'

**EXPORT\_TO** = ['text/xml:CTS', 'application/ld+json:DTS', 'text/xml:CTS\_CapiTainS']

**MODEL\_URI** = rdflib.term.URIRef('https://w3id.org/dts/api#collection')

**TYPE\_URI** = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/CtsTextInventoryCollection')

**asNode** ()

Node representation of the collection in the graph

**Return type** URIRef

**children**

Dictionary of childrens {Identifier: Collection}

**Return type** `dict`

**descendants**

Any descendant (no max level) of the collection's item

**Return type** `[Collection]`

**export** (*output=None, \*\*kwargs*)

Export the collection item in the Mimetype required.

**Parameters** **output** (*str*) – Mimetype to export to (Uses MyCapytain.common.utils.Mimetypes)

**Returns** Object using a different representation

**classmethod export\_base\_dts** (*graph, obj, nsm*)

Export the base DTS information in a simple reusable way

**Parameters**

- **graph** – Current graph where the information lie
- **obj** – Object for which we build info
- **nsm** – Namespace manager

**Returns** `Dict`

**export\_capacities**

List Mimetypes that current object can export to

**get\_cts\_property** (*prop, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to get (Without namespace)
- **lang** – Language to get for given value

**Returns** Value or default if lang is set, else whole set of values

**Return type** `dict` or `Literal`

**get\_label** (*lang=None*)

Return label for given lang or any default

**Parameters** **lang** – Language to request

**Returns** Label value

**Return type** `Literal`

**get\_link** (*prop*)

Get given link in CTS Namespace

**Parameters** **prop** – Property to get (Without namespace)

**Returns** whole set of values

**Return type** `list`

**graph**

RDFLib Graph space

**Return type** `Graph`

**id**

**members**

Children of the collection's item

**Return type** [Collection]

**metadata**

**model**

**parent**

Parent of current object

**Return type** Collection

**parents**

Iterator to find parents of current collection, from closest to furthest

**Return type** Generator[Collection]

**readable**

Readable property should return elements where the element can be queried for `getPassage / getReffs`

**readableDescendants**

List of element available which are readable

**Return type** [Collection]

**set\_cts\_property** (*prop, value, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property
- **lang** – Language to set for given value

**set\_label** (*label, lang*)

Add the label of the collection in given lang

**Parameters**

- **label** – Label Value
- **lang** – Language code

**set\_link** (*prop, value*)

Set given link in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property

**size**

**type**

**urn**

**version**

```
class MyCapytain.resources.prototypes.cts.inventory.CtsEditionMetadata (urn=",
                                                                    par-
                                                                    ent=None,
                                                                    lang=None)
```

Bases: MyCapytain.resources.prototypes.cts.inventory.CtsTextMetadata

Represents a CTS XmlCtsEditionMetadata

#### Parameters

- **urn** (*str*) – Identifier of the CtsTextMetadata
- **parent** (*CtsWorkMetadata*) – Parent of current item

```
CTS_LINKS = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/about')]
```

```
CTS_PROPERTIES = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/label'), rdflib
```

```
DC_TITLE_KEY = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/label')
```

```
DEFAULT_EXPORT = 'python/lxml'
```

```
EXPORT_TO = ['text/xml:CTS', 'text/xml:CTS_CapitainS']
```

```
MODEL_URI = rdflib.term.URIRef('https://w3id.org/dts/api#resource')
```

```
SUBTYPE = 'edition'
```

```
TYPE_URI = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/edition')
```

```
asNode()
```

Node representation of the collection in the graph

**Return type** URIRef

```
children
```

Dictionary of childrens {Identifier: Collection}

**Return type** dict

```
descendants
```

Descendants of the collection's item

**Warning:** CapitainsCtsText has no Descendants

**Return type** list

```
editions()
```

Get all editions of the texts

**Returns** List of editions

**Return type** [CtsTextMetadata]

```
export (output=None, **kwargs)
```

Export the collection item in the Mimetype required.

**Parameters** **output** (*str*) – Mimetype to export to (Uses MyCapytain.common.utils.Mimetypes)

**Returns** Object using a different representation

```
classmethod export_base_dts (graph, obj, nsm)
```

Export the base DTS information in a simple reusable way

**Parameters**

- **graph** – Current graph where the information lie
- **obj** – Object for which we build info
- **nsm** – Namespace manager

**Returns** Dict

**export\_capacities**

List Mimetypes that current object can export to

**get\_creator** (*lang=None*)

Get the DC Creator literal value

**Parameters** **lang** – Language to retrieve

**Returns** Creator string representation

**Return type** Literal

**get\_cts\_property** (*prop, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to get (Without namespace)
- **lang** – Language to get for given value

**Returns** Value or default if lang is set, else whole set of values

**Return type** dict or Literal

**get\_description** (*lang=None*)

Get the DC description of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Description string representation

**Return type** Literal

**get\_label** (*lang=None*)

Return label for given lang or any default

**Parameters** **lang** – Language to request

**Returns** Label value

**Return type** Literal

**get\_link** (*prop*)

Get given link in CTS Namespace

**Parameters** **prop** – Property to get (Without namespace)

**Returns** whole set of values

**Return type** list

**get\_subject** (*lang=None*)

Get the DC subject of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Subject string representation

**Return type** Literal

**get\_title** (*lang=None*)

Get the DC Title of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Title string representation

**Return type** Literal

**graph**

RDFLib Graph space

**Return type** Graph

**id**

**lang**

Languages this text is in

**Returns** List of available languages

**members**

Children of the collection's item

**Warning:** CapitainsCtsText has no children

**Return type** [list](#)

**metadata**

**model**

**parent**

Parent of current object

**Return type** Collection

**parents**

Iterator to find parents of current collection, from closest to furthest

**Return type** Generator[Collection]

**readable**

Readable property should return elements where the element can be queried for `getPassage` / `getReffs`

**readableDescendants**

List of element available which are readable

**Return type** [Collection]

**set\_cts\_property** (*prop, value, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property
- **lang** – Language to set for given value

**set\_label** (*label, lang*)

Add the label of the collection in given lang

**Parameters**

- **label** – Label Value
- **lang** – Language code

**set\_link** (*prop, value*)

Set given link in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property

**size****subtype**

Subtype of the object

**Returns** string representation of subtype**translations** (*key=None*)

Get translations in given language

**Parameters** **key** – Language ISO Code to filter on**Returns****type****urn****version**

```
class MyCapytain.resources.prototypes.cts.inventory.CtsWorkMetadata (urn=None,  
                                                                    par-  
                                                                    ent=None)
```

Bases: MyCapytain.resources.prototypes.cts.inventory.PrototypeCtsCollection

Represents a CTS CtsWorkMetadata

CTS CtsWorkMetadata can be added to each other which would most likely happen if you take your data from multiple API or Textual repository. This works close to dictionary update in Python. See update

**Parameters**

- **urn** (URN) – Identifier of the CtsWorkMetadata
- **parent** (*CtsTextgroupMetadata*) – Parent of current object

**Variables** **urn** – URN Identifier**CTS\_LINKS** = []**CTS\_PROPERTIES** = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/title')]**DC\_TITLE\_KEY** = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/title')**DEFAULT\_EXPORT** = 'python/lxml'**EXPORT\_TO** = ['text/xml:CTS', 'text/xml:CTS\_CapiTainS']**MODEL\_URI** = rdflib.term.URIRef('https://w3id.org/dts/api#collection')**TYPE\_URI** = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/work')**asNode** ()

Node representation of the collection in the graph

**Return type** URIRef

**children**

Dictionary of childrens {Identifier: Collection}

**Return type** dict

**descendants**

Any descendant (no max level) of the collection's item

**Return type** [Collection]

**export** (*output=None, \*\*kwargs*)

Export the collection item in the Mimetype required.

**Parameters** **output** (*str*) – Mimetype to export to (Uses MyCapytain.common.utils.Mimetypes)

**Returns** Object using a different representation

**classmethod export\_base\_dts** (*graph, obj, nsm*)

Export the base DTS information in a simple reusable way

**Parameters**

- **graph** – Current graph where the information lie
- **obj** – Object for which we build info
- **nsm** – Namespace manager

**Returns** Dict

**export\_capacities**

List Mimetypes that current object can export to

**get\_cts\_property** (*prop, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to get (Without namespace)
- **lang** – Language to get for given value

**Returns** Value or default if lang is set, else whole set of values

**Return type** dict or Literal

**get\_label** (*lang=None*)

Return label for given lang or any default

**Parameters** **lang** – Language to request

**Returns** Label value

**Return type** Literal

**get\_link** (*prop*)

Get given link in CTS Namespace

**Parameters** **prop** – Property to get (Without namespace)

**Returns** whole set of values

**Return type** list

**get\_translation\_in** (*key=None*)

Find a translation with given language

**Parameters** **key** (*text\_type*) – Language to find

**Return type** [CtsTextMetadata]

**Returns** List of available translations

**graph**

RDFLib Graph space

**Return type** Graph

**id**

**lang**

Languages this text is in

**Returns** List of available languages

**members**

Children of the collection's item

**Return type** [Collection]

**metadata**

**model**

**parent**

Parent of current object

**Return type** Collection

**parents**

Iterator to find parents of current collection, from closest to furthest

**Return type** Generator[Collection]

**readable**

Readable property should return elements where the element can be queried for `getPassage / getReffs`

**readableDescendants**

List of element available which are readable

**Return type** [Collection]

**set\_cts\_property** (*prop, value, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property
- **lang** – Language to set for given value

**set\_label** (*label, lang*)

Add the label of the collection in given lang

**Parameters**

- **label** – Label Value
- **lang** – Language code

**set\_link** (*prop, value*)

Set given link in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property

**size**

**texts**

Texts

**Returns** Dictionary of texts

**Return type** defaultdict(PrototypeTexts)

**type**

**update** (*other*)

Merge two XmlCtsWorkMetadata Objects.

- Original (left Object) keeps his parent.
- Added document overwrite text if it already exists

**Parameters** **other** (*CtsWorkMetadata*) – XmlCtsWorkMetadata object

**Returns** XmlCtsWorkMetadata Object

**Rtype** XmlCtsWorkMetadata

**urn**

**version**

```
class MyCapytain.resources.prototypes.cts.inventory.CtsCommentaryMetadata (urn="",
                                                                    par-
                                                                    ent=None,
                                                                    lang=None)
```

Bases: MyCapytain.resources.prototypes.cts.inventory.CtsTextMetadata

Represents a CTS Commentary

**Parameters**

- **urn** (*str*) – Identifier of the PrototypeText
- **parent** (*PrototypeWork*) – Parent of current item
- **lang** (*Lang*) – Language of the commentary

```
CTS_LINKS = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/about')]
```

```
CTS_PROPERTIES = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/label'), rdflib
```

```
DC_TITLE_KEY = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/label')
```

```
DEFAULT_EXPORT = 'python/lxml'
```

```
EXPORT_TO = ['text/xml:CTS', 'text/xml:CTS_CapiTainS']
```

```
MODEL_URI = rdflib.term.URIRef('https://w3id.org/dts/api#resource')
```

```
SUBTYPE = 'commentary'
```

```
TYPE_URI = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/commentary')
```

**asNode** ()

Node representation of the collection in the graph

**Return type** URIRef

**children**

Dictionary of childrens {Identifier: Collection}

**Return type** dict

**descendants**

Descendants of the collection's item

**Warning:** CapitainsCtsText has no Descendants

**Return type** list

**editions** ()

Get all editions of the texts

**Returns** List of editions

**Return type** [CtsTextMetadata]

**export** (*output=None, \*\*kwargs*)

Export the collection item in the Mimetype required.

**Parameters** **output** (*str*) – Mimetype to export to (Uses MyCapytain.common.utils.Mimetypes)

**Returns** Object using a different representation

**classmethod export\_base\_dts** (*graph, obj, nsm*)

Export the base DTS information in a simple reusable way

**Parameters**

- **graph** – Current graph where the information lie
- **obj** – Object for which we build info
- **nsm** – Namespace manager

**Returns** Dict

**export\_capacities**

List Mimetypes that current object can export to

**get\_creator** (*lang=None*)

Get the DC Creator literal value

**Parameters** **lang** – Language to retrieve

**Returns** Creator string representation

**Return type** Literal

**get\_cts\_property** (*prop, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to get (Without namespace)
- **lang** – Language to get for given value

**Returns** Value or default if lang is set, else whole set of values

**Return type** dict or Literal

**get\_description** (*lang=None*)

Get the DC description of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Description string representation

**Return type** Literal

**get\_label** (*lang=None*)

Return label for given lang or any default

**Parameters** **lang** – Language to request

**Returns** Label value

**Return type** Literal

**get\_link** (*prop*)

Get given link in CTS Namespace

**Parameters** **prop** – Property to get (Without namespace)

**Returns** whole set of values

**Return type** list

**get\_subject** (*lang=None*)

Get the DC subject of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Subject string representation

**Return type** Literal

**get\_title** (*lang=None*)

Get the DC Title of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Title string representation

**Return type** Literal

**graph**

RDFLib Graph space

**Return type** Graph

**id**

**lang**

Languages this text is in

**Returns** List of available languages

**members**

Children of the collection's item

|  |
|--|
| <b>Warning:</b> CapitainsCtsText has no children |
|--|

**Return type** *list*

**metadata**

**model**

**parent**

Parent of current object

**Return type** *Collection*

**parents**

Iterator to find parents of current collection, from closest to furthest

**Return type** *Generator[Collection]*

**readable**

Readable property should return elements where the element can be queried for *getPassage / getReffs*

**readableDescendants**

List of element available which are readable

**Return type** *[Collection]*

**set\_cts\_property** (*prop, value, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property
- **lang** – Language to set for given value

**set\_label** (*label, lang*)

Add the label of the collection in given lang

**Parameters**

- **label** – Label Value
- **lang** – Language code

**set\_link** (*prop, value*)

Set given link in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property

**size**

**subtype**

Subtype of the object

**Returns** *string representation of subtype*

**translations** (*key=None*)

Get translations in given language

**Parameters** **key** – Language ISO Code to filter on

**Returns**

**type**

**urn****version**

```
class MyCapytain.resources.prototypes.cts.inventory.CtsTextgroupMetadata (urn=",
                                                                    par-
                                                                    ent=None)
```

Bases: MyCapytain.resources.prototypes.cts.inventory.PrototypeCtsCollection

Represents a CTS Textgroup

CTS CtsTextgroupMetadata can be added to each other which would most likely happen if you take your data from multiple API or Textual repository. This works close to dictionary update in Python. See update

#### Parameters

- **urn** (URN) – Identifier of the CtsTextgroupMetadata
- **parent** (CtsTextInventoryMetadata) – Parent of the current object

**Variables** *urn* – URN Identifier

```
CTS_LINKS = []
```

```
CTS_PROPERTIES = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/groupname')]
```

```
DC_TITLE_KEY = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/groupname')
```

```
DEFAULT_EXPORT = 'python/lxml'
```

```
EXPORT_TO = ['text/xml:CTS', 'text/xml:CTS_CapiTainS']
```

```
MODEL_URI = rdflib.term.URIRef('https://w3id.org/dts/api#collection')
```

```
TYPE_URI = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/textgroup')
```

```
asNode()
```

Node representation of the collection in the graph

**Return type** URIRef

**children**

Dictionary of childrens {Identifier: Collection}

**Return type** dict

**descendants**

Any descendant (no max level) of the collection's item

**Return type** [Collection]

**export** (output=None, \*\*kwargs)

Export the collection item in the Mimetype required.

**Parameters** *output* (*str*) – Mimetype to export to (Uses MyCapytain.common.utils.Mimetypes)

**Returns** Object using a different representation

**classmethod export\_base\_dts** (graph, obj, nsm)

Export the base DTS information in a simple reusable way

**Parameters**

- **graph** – Current graph where the information lie
- **obj** – Object for which we build info
- **nsm** – Namespace manager

**Returns** Dict

**export\_capacities**

List Mimetypes that current object can export to

**get\_cts\_property** (*prop*, *lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to get (Without namespace)
- **lang** – Language to get for given value

**Returns** Value or default if lang is set, else whole set of values

**Return type** dict or Literal

**get\_label** (*lang=None*)

Return label for given lang or any default

**Parameters** **lang** – Language to request

**Returns** Label value

**Return type** Literal

**get\_link** (*prop*)

Get given link in CTS Namespace

**Parameters** **prop** – Property to get (Without namespace)

**Returns** whole set of values

**Return type** list

**graph**

RDFLib Graph space

**Return type** Graph

**id**

**members**

Children of the collection's item

**Return type** [Collection]

**metadata**

**model**

**parent**

Parent of current object

**Return type** Collection

**parents**

Iterator to find parents of current collection, from closest to furthest

**Return type** Generator[Collection]

**readable**

Readable property should return elements where the element can be queried for getPassage / getReffs

**readableDescendants**

List of element available which are readable

**Return type** [Collection]

**set\_cts\_property** (*prop, value, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property
- **lang** – Language to set for given value

**set\_label** (*label, lang*)

Add the label of the collection in given lang

**Parameters**

- **label** – Label Value
- **lang** – Language code

**set\_link** (*prop, value*)

Set given link in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property

**size**

**type**

**update** (*other*)

Merge two Textgroup Objects.

- Original (left Object) keeps his parent.
- Added document merges with work if it already exists

**Parameters** **other** (*CtsTextgroupMetadata*) – Textgroup object

**Returns** Textgroup Object

**Return type** CtsTextgroupMetadata

**urn**

**version**

**works**

Works

**Returns** Dictionary of works

**Return type** defaultdict(PrototypeWorks)

**class** MyCapytain.resources.prototypes.cts.inventory.CtsTextInventoryMetadata (*name='defaultInven*

*parent=None*)

Bases: MyCapytain.resources.prototypes.cts.inventory.PrototypeCtsCollection

Initiate a CtsTextInventoryMetadata resource

**Parameters**

- **resource** (*Any*) – Resource representing the CtsTextInventoryMetadata

- **name** (*str*) – Identifier of the CtsTextInventoryMetadata

**CTS\_LINKS** = []

**CTS\_PROPERTIES** = []

**DC\_TITLE\_KEY** = `rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/name')`

**DEFAULT\_EXPORT** = 'python/lxml'

**EXPORT\_TO** = ['text/xml:CTS', 'text/xml:CTS\_CapiTainS']

**MODEL\_URI** = `rdflib.term.URIRef('https://w3id.org/dts/api#collection')`

**TYPE\_URI** = `rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/TextInventory')`

**asNode** ()

Node representation of the collection in the graph

**Return type** URIRef

**children**

Dictionary of childrens {Identifier: Collection}

**Return type** dict

**descendants**

Any descendant (no max level) of the collection's item

**Return type** [Collection]

**export** (*output=None, \*\*kwargs*)

Export the collection item in the Mime type required.

**Parameters** **output** (*str*) – Mime type to export to (Uses MyCapytain.common.utils.Mimetypes)

**Returns** Object using a different representation

**classmethod export\_base\_dts** (*graph, obj, nsm*)

Export the base DTS information in a simple reusable way

**Parameters**

- **graph** – Current graph where the information lie
- **obj** – Object for which we build info
- **nsm** – Namespace manager

**Returns** Dict

**export\_capacities**

List Mimetypes that current object can export to

**get\_cts\_property** (*prop, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to get (Without namespace)
- **lang** – Language to get for given value

**Returns** Value or default if lang is set, else whole set of values

**Return type** dict or Literal

**get\_label** (*lang=None*)

Return label for given lang or any default

**Parameters** **lang** – Language to request

**Returns** Label value

**Return type** Literal

**get\_link** (*prop*)

Get given link in CTS Namespace

**Parameters** **prop** – Property to get (Without namespace)

**Returns** whole set of values

**Return type** list

**graph**

RDFLib Graph space

**Return type** Graph

**id**

**members**

Children of the collection's item

**Return type** [Collection]

**metadata**

**model**

**parent**

Parent of current object

**Return type** Collection

**parents**

Iterator to find parents of current collection, from closest to furthest

**Return type** Generator[Collection]

**readable**

Readable property should return elements where the element can be queried for getPassage / getReffs

**readableDescendants**

List of element available which are readable

**Return type** [Collection]

**set\_cts\_property** (*prop, value, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property
- **lang** – Language to set for given value

**set\_label** (*label, lang*)

Add the label of the collection in given lang

**Parameters**

- **label** – Label Value
- **lang** – Language code

**set\_link** (*prop, value*)

Set given link in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property

**size**

**textgroups**

Textgroups

**Returns** Dictionary of textgroups

**Return type** defaultdict(CtsTextgroupMetadata)

**type**

**urn**

**version**

```
class MyCapytain.resources.prototypes.cts.inventory.CtsTextMetadata (urn="",
                                                                    par-
                                                                    ent=None,
                                                                    lang=None)
```

Bases: MyCapytain.resources.prototypes.metadata.ResourceCollection,  
MyCapytain.resources.prototypes.cts.inventory.PrototypeCtsCollection

Represents a CTS CtsTextMetadata

**Parameters**

- **urn** (URN) – Identifier of the CtsTextMetadata
- **parent** ([PrototypeCtsCollection]) – Item parents of the current collection

**Variables** *urn* – URN Identifier

```
CTS_LINKS = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/about')]
```

```
CTS_PROPERTIES = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/label'), rdflib
```

```
DC_TITLE_KEY = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/label')
```

```
DEFAULT_EXPORT = 'python/lxml'
```

```
EXPORT_TO = ['text/xml:CTS', 'text/xml:CTS_CapiTainS']
```

```
MODEL_URI = rdflib.term.URIRef('https://w3id.org/dts/api#resource')
```

```
SUBTYPE = 'unknown'
```

```
TYPE_URI = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/text')
```

```
asNode ()
```

Node representation of the collection in the graph

**Return type** URIRef

**children**

Dictionary of childrens {Identifier: Collection}

**Return type** `dict`

**descendants**

Descendants of the collection's item

**Warning:** CapitainsCtsText has no Descendants

**Return type** `list`

**editions** ()

Get all editions of the texts

**Returns** List of editions

**Return type** `[CtsTextMetadata]`

**export** (*output=None, \*\*kwargs*)

Export the collection item in the Mime type required.

**Parameters** **output** (*str*) – Mime type to export to (Uses `MyCapytain.common.utils.Mimetypes`)

**Returns** Object using a different representation

**classmethod export\_base\_dts** (*graph, obj, nsm*)

Export the base DTS information in a simple reusable way

**Parameters**

- **graph** – Current graph where the information lie
- **obj** – Object for which we build info
- **nsm** – Namespace manager

**Returns** Dict

**export\_capacities**

List Mime types that current object can export to

**get\_creator** (*lang=None*)

Get the DC Creator literal value

**Parameters** **lang** – Language to retrieve

**Returns** Creator string representation

**Return type** Literal

**get\_cts\_property** (*prop, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to get (Without namespace)
- **lang** – Language to get for given value

**Returns** Value or default if lang is set, else whole set of values

**Return type** `dict` or Literal

**get\_description** (*lang=None*)

Get the DC description of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Description string representation

**Return type** Literal

**get\_label** (*lang=None*)

Return label for given lang or any default

**Parameters** **lang** – Language to request

**Returns** Label value

**Return type** Literal

**get\_link** (*prop*)

Get given link in CTS Namespace

**Parameters** **prop** – Property to get (Without namespace)

**Returns** whole set of values

**Return type** *list*

**get\_subject** (*lang=None*)

Get the DC subject of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Subject string representation

**Return type** Literal

**get\_title** (*lang=None*)

Get the DC Title of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Title string representation

**Return type** Literal

**graph**

RDFLib Graph space

**Return type** Graph

**id**

**lang**

Languages this text is in

**Returns** List of available languages

**members**

Children of the collection's item

**Warning:** CapitainsCtsText has no children

**Return type** *list*

**metadata**

**model**

**parent**

Parent of current object

**Return type** Collection

**parents**

Iterator to find parents of current collection, from closest to furthest

**Return type** Generator[Collection]

**readable**

Readable property should return elements where the element can be queried for `getPassage / getReffs`

**readableDescendants**

List of element available which are readable

**Return type** [Collection]

**set\_cts\_property** (*prop, value, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property
- **lang** – Language to set for given value

**set\_label** (*label, lang*)

Add the label of the collection in given lang

**Parameters**

- **label** – Label Value
- **lang** – Language code

**set\_link** (*prop, value*)

Set given link in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property

**size****subtype**

Subtype of the object

**Returns** string representation of subtype

**translations** (*key=None*)

Get translations in given language

**Parameters** **key** – Language ISO Code to filter on

**Returns**

**type****urn****version**

```
class MyCapytain.resources.prototypes.cts.inventory.CtsTranslationMetadata (urn=",
                                                                    par-
                                                                    ent=None,
                                                                    lang=None)
```

Bases: MyCapytain.resources.prototypes.cts.inventory.CtsTextMetadata

Represents a CTS XmlCtsTranslationMetadata

**Parameters**

- **urn** (*str*) – Identifier of the CtsTextMetadata
- **parent** (*CtsWorkMetadata*) – Parent of current item
- **lang** (*Lang*) – Language of the translation

```
CTS_LINKS = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/about')]
```

```
CTS_PROPERTIES = [rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/label'), rdflib
```

```
DC_TITLE_KEY = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/label')
```

```
DEFAULT_EXPORT = 'python/lxml'
```

```
EXPORT_TO = ['text/xml:CTS', 'text/xml:CTS_CapiTainS']
```

```
MODEL_URI = rdflib.term.URIRef('https://w3id.org/dts/api#resource')
```

```
SUBTYPE = 'translation'
```

```
TYPE_URI = rdflib.term.URIRef('http://chs.harvard.edu/xmlns/cts/translation')
```

```
asNode()
```

Node representation of the collection in the graph

**Return type** URIRef

**children**

Dictionary of childrens {Identifier: Collection}

**Return type** dict

**descendants**

Descendants of the collection's item

**Warning:** CapitainsCtsText has no Descendants

**Return type** list

```
editions()
```

Get all editions of the texts

**Returns** List of editions

**Return type** [CtsTextMetadata]

```
export (output=None, **kwargs)
```

Export the collection item in the Mimetype required.

**Parameters** **output** (*str*) – Mimetype to export to (Uses MyCapytain.common.utils.Mimetypes)

**Returns** Object using a different representation

**classmethod export\_base\_dts** (*graph, obj, nsm*)

Export the base DTS information in a simple reusable way

**Parameters**

- **graph** – Current graph where the information lie
- **obj** – Object for which we build info
- **nsm** – Namespace manager

**Returns** Dict

**export\_capacities**

List Mimetypes that current object can export to

**get\_creator** (*lang=None*)

Get the DC Creator literal value

**Parameters lang** – Language to retrieve

**Returns** Creator string representation

**Return type** Literal

**get\_cts\_property** (*prop, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to get (Without namespace)
- **lang** – Language to get for given value

**Returns** Value or default if lang is set, else whole set of values

**Return type** dict or Literal

**get\_description** (*lang=None*)

Get the DC description of the object

**Parameters lang** – Lang to retrieve

**Returns** Description string representation

**Return type** Literal

**get\_label** (*lang=None*)

Return label for given lang or any default

**Parameters lang** – Language to request

**Returns** Label value

**Return type** Literal

**get\_link** (*prop*)

Get given link in CTS Namespace

**Parameters prop** – Property to get (Without namespace)

**Returns** whole set of values

**Return type** list

**get\_subject** (*lang=None*)

Get the DC subject of the object

**Parameters lang** – Lang to retrieve

**Returns** Subject string representation

**Return type** Literal

**get\_title** (*lang=None*)

Get the DC Title of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Title string representation

**Return type** Literal

**graph**

RDFLib Graph space

**Return type** Graph

**id**

**lang**

Languages this text is in

**Returns** List of available languages

**members**

Children of the collection's item

**Warning:** CapitainsCtsText has no children

**Return type** [list](#)

**metadata**

**model**

**parent**

Parent of current object

**Return type** Collection

**parents**

Iterator to find parents of current collection, from closest to furthest

**Return type** Generator[Collection]

**readable**

Readable property should return elements where the element can be queried for `getPassage / getReffs`

**readableDescendants**

List of element available which are readable

**Return type** [Collection]

**set\_cts\_property** (*prop, value, lang=None*)

Set given property in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property
- **lang** – Language to set for given value

**set\_label** (*label, lang*)

Add the label of the collection in given lang

**Parameters**

- **label** – Label Value
- **lang** – Language code

**set\_link** (*prop, value*)

Set given link in CTS Namespace

**Parameters**

- **prop** – Property to set (Without namespace)
- **value** – Value to set for given property

**size**

**subtype**

Subtype of the object

**Returns** string representation of subtype

**translations** (*key=None*)

Get translations in given language

**Parameters** **key** – Language ISO Code to filter on

**Returns**

**type**

**urn**

**version**

## Text Prototypes

```
class MyCapytain.resources.prototypes.text.TextualElement (identifier: str = None,
                                                         metadata: MyCapytain.common.metadata.Metadata
                                                         = None)
```

Bases: MyCapytain.common.base.Exportable

Node representing a text passage.

**Parameters**

- **identifier** (*str*) – Identifier of the text
- **metadata** (*Collection*) – Collection Information about the Item

**Variables** **default\_exclude** – Default exclude for exports

**DEFAULT\_EXPORT** = None

**EXPORT\_TO** = []

**asNode** () → rdflib.term.Identifier

**default\_exclude** = []

**export** (*output: str = None, exclude: List[str] = None, \*\*kwargs*)

Export the collection item in the Mimetype required.

..note:: If current implementation does not have special mimetypes, reuses default\_export method

**Parameters**

- **output** (*str*) – Mimetype to export to (Uses MyCapytain.common.constants.Mimetypes)
- **exclude** (*[str]*) – Information to exclude. Specific to implementations

**Returns** Object using a different representation

**export\_capacities**

List Mimetypes that current object can export to

**get\_creator** (*lang: str = None*) → rdflib.term.Literal

Get the DC Creator literal value

**Parameters** **lang** – Language to retrieve

**Returns** Creator string representation

**Return type** Literal

**get\_description** (*lang: str = None*) → rdflib.term.Literal

Get the description of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Description string representation

**Return type** Literal

**get\_subject** (*lang=None*) → rdflib.term.Literal

Get the subject of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Subject string representation

**Return type** Literal

**get\_title** (*lang: str = None*) → rdflib.term.Literal

Get the title of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Title string representation

**Return type** Literal

**graph**

**id**

Identifier of the text

**Returns** Identifier of the text

**metadata**

Metadata information about the text

**Returns** Collection object with metadata about the text

**Return type** Metadata

**set\_creator** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)  
Set the DC Creator literal value

**Parameters**

- **value** – Value of the creator node
- **lang** – Language in which the value is

**set\_description** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)  
Set the DC Description literal value

**Parameters**

- **value** – Value of the title node
- **lang** – Language in which the value is

**set\_subject** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)  
Set the DC Subject literal value

**Parameters**

- **value** – Value of the subject node
- **lang** – Language in which the value is

**set\_title** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)  
Set the DC Title literal value

**Parameters**

- **value** – Value of the title node
- **lang** – Language in which the value is

**text**

String representation of the text

**Returns** String representation of the text

**class** MyCapytain.resources.prototypes.text.**TextualGraph** (*identifier: str = None, \*\*kwargs*)

Bases: MyCapytain.resources.prototypes.text.TextualNode

Node representing a text passage.

**Parameters**

- **identifier** (*str*) – Identifier of the text
- **metadata** (*Collection*) – Collection Information about the Item
- **citation** (*Citation*) – XmlCtsCitation system of the text
- **children** (*[str]*) – Current node Children’s Identifier
- **parent** (*str*) – Parent of the current node
- **siblings** (*str*) – Previous and next node of the current node
- **depth** (*int*) – Depth of the node in the global hierarchy of the text tree
- **resource** – Resource used to navigate through the textual graph

**Variables** `default_exclude` – Default exclude for exports

`DEFAULT_EXPORT = None`

`EXPORT_TO = []`

**asNode** () → rdflib.term.Identifier

**childIds**  
Children Ids

**citation**  
Citation system of the object

**Return type** *Citation*

**default\_exclude** = []

**depth**  
Depth of the node in the global hierarchy of the text tree

**export** (*output: str = None, exclude: List[str] = None, \*\*kwargs*)  
Export the collection item in the Mimetype required.

..note:: If current implementation does not have special mimetypes, reuses default\_export method

**Parameters**

- **output** (*str*) – Mimetype to export to (Uses MyCapytain.common.constants.Mimetypes)
- **exclude** (*[str]*) – Information to exclude. Specific to implementations

**Returns** Object using a different representation

**export\_capacities**  
List Mimetypes that current object can export to

**firstId**  
First child Id

**getReffs** (*level: int = 1, subreference: MyCapytain.common.reference.\_base.BaseReference = None*)  
→ MyCapytain.common.reference.\_base.BaseReferenceSet  
CtsReference available at a given level

**Parameters**

- **level** – Depth required. If not set, should retrieve first encountered level (1 based)
- **subreference** – Subreference (optional)

**Returns** List of levels

**getTextualNode** (*subreference: MyCapytain.common.reference.\_base.BaseReference*) → MyCapytain.resources.prototypes.text.TextualGraph  
Retrieve a passage and store it in the object

**Parameters** **subreference** – CtsReference of the passage to retrieve

**Returns** Object representing the passage

**Raises** *TypeError* when reference is not a list or a CtsReference

**get\_creator** (*lang: str = None*) → rdflib.term.Literal  
Get the DC Creator literal value

**Parameters** **lang** – Language to retrieve

**Returns** Creator string representation

**Return type** Literal

**get\_description** (*lang: str = None*) → rdflib.term.Literal  
Get the description of the object

**Parameters** `lang` – Lang to retrieve

**Returns** Description string representation

**Return type** Literal

**get\_subject** (*lang=None*) → rdflib.term.Literal

Get the subject of the object

**Parameters** `lang` – Lang to retrieve

**Returns** Subject string representation

**Return type** Literal

**get\_title** (*lang: str = None*) → rdflib.term.Literal

Get the title of the object

**Parameters** `lang` – Lang to retrieve

**Returns** Title string representation

**Return type** Literal

**graph**

**id**

Identifier of the text

**Returns** Identifier of the text

**lastId**

Last child id

**metadata**

Metadata information about the text

**Returns** Collection object with metadata about the text

**Return type** Metadata

**nextId**

Next Id

**parentId**

Parent Id

**prevId**

Previous Id (Sibling)

**set\_creator** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)

Set the DC Creator literal value

**Parameters**

- **value** – Value of the creator node
- **lang** – Language in which the value is

**set\_description** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)

Set the DC Description literal value

**Parameters**

- **value** – Value of the title node
- **lang** – Language in which the value is

**set\_subject** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str]*, *lang: str = None*)  
 Set the DC Subject literal value

**Parameters**

- **value** – Value of the subject node
- **lang** – Language in which the value is

**set\_title** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str]*, *lang: str = None*)  
 Set the DC Title literal value

**Parameters**

- **value** – Value of the title node
- **lang** – Language in which the value is

**siblingsId**  
 Siblings Id

**text**  
 String representation of the text

**Returns** String representation of the text

```
class MyCapytain.resources.prototypes.text.TextualNode (identifier: str = None,  

citation: MyCapytain.common.reference._capitains_cts.Citation  

= None, **kwargs)
```

Bases: MyCapytain.resources.prototypes.text.TextualElement, MyCapytain.common.reference.\_base.NodeId

Node representing a text passage.

**Parameters**

- **identifier** (*str*) – Identifier of the text
- **metadata** (*Collection*) – Collection Information about the Item
- **citation** (*Citation*) – XmlCtsCitation system of the text
- **children** (*[str]*) – Current node Children’s Identifier
- **parent** (*str*) – Parent of the current node
- **siblings** (*str*) – Previous and next node of the current node
- **depth** (*int*) – Depth of the node in the global hierarchy of the text tree

**Variables** *default\_exclude* – Default exclude for exports

**DEFAULT\_EXPORT** = None

**EXPORT\_TO** = []

**asNode** () → rdflib.term.Identifier

**childIds**  
 Children Ids

**citation**  
 Citation system of the object

**Return type** *Citation*

**default\_exclude** = []

**depth**

Depth of the node in the global hierarchy of the text tree

**export** (*output: str = None, exclude: List[str] = None, \*\*kwargs*)

Export the collection item in the Mimetype required.

..note:: If current implementation does not have special mimetypes, reuses default\_export method

**Parameters**

- **output** (*str*) – Mimetype to export to (Uses MyCapytain.common.constants.Mimetypes)
- **exclude** (*[str]*) – Information to exclude. Specific to implementations

**Returns** Object using a different representation

**export\_capacities**

List Mimitypes that current object can export to

**firstId**

First child Id

**get\_creator** (*lang: str = None*) → rdflib.term.Literal

Get the DC Creator literal value

**Parameters** **lang** – Language to retrieve

**Returns** Creator string representation

**Return type** Literal

**get\_description** (*lang: str = None*) → rdflib.term.Literal

Get the description of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Description string representation

**Return type** Literal

**get\_subject** (*lang=None*) → rdflib.term.Literal

Get the subject of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Subject string representation

**Return type** Literal

**get\_title** (*lang: str = None*) → rdflib.term.Literal

Get the title of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Title string representation

**Return type** Literal

**graph****id**

Identifier of the text

**Returns** Identifier of the text

**lastId**

Last child id

**metadata**

Metadata information about the text

**Returns** Collection object with metadata about the text

**Return type** Metadata

**nextId**

Next Id

**parentId**

Parent Id

**prevId**

Previous Id (Sibling)

**set\_creator** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)

Set the DC Creator literal value

**Parameters**

- **value** – Value of the creator node
- **lang** – Language in which the value is

**set\_description** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)

Set the DC Description literal value

**Parameters**

- **value** – Value of the title node
- **lang** – Language in which the value is

**set\_subject** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)

Set the DC Subject literal value

**Parameters**

- **value** – Value of the subject node
- **lang** – Language in which the value is

**set\_title** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)

Set the DC Title literal value

**Parameters**

- **value** – Value of the title node
- **lang** – Language in which the value is

**siblingsId**

Siblings Id

**text**

String representation of the text

**Returns** String representation of the text

**class** MyCapytain.resources.prototypes.text.**InteractiveTextualNode** (*identifier: str = None, \*\*kwargs*)

Bases: MyCapytain.resources.prototypes.text.TextualGraph

Node representing a text passage.

**Parameters**

- **identifier** (*str*) – Identifier of the text
- **metadata** (*Collection*) – Collection Information about the Item
- **citation** (*Citation*) – XmlCtsCitation system of the text
- **children** (*[str]*) – Current node Children’s Identifier
- **parent** (*str*) – Parent of the current node
- **siblings** (*str*) – Previous and next node of the current node
- **depth** (*int*) – Depth of the node in the global hierarchy of the text tree
- **resource** – Resource used to navigate through the textual graph

Variables `default_exclude` – Default exclude for exports

`DEFAULT_EXPORT = None`

`EXPORT_TO = []`

`asNode ()` → `rdflib.term.Identifier`

**childIds**

Identifiers of children

**Returns** Identifiers of children

**children**

Children TextualNode

**citation**

Citation system of the object

**Return type** *Citation*

`default_exclude = []`

**depth**

Depth of the node in the global hierarchy of the text tree

**export** (*output: str = None, exclude: List[str] = None, \*\*kwargs*)

Export the collection item in the Mimetype required.

..note:: If current implementation does not have special mimetypes, reuses default\_export method

**Parameters**

- **output** (*str*) – Mimetype to export to (Uses `MyCapytain.common.constants.Mimetypes`)
- **exclude** (*[str]*) – Information to exclude. Specific to implementations

**Returns** Object using a different representation

**export\_capacities**

List Mimetypes that current object can export to

**first**

First TextualNode

**firstId**

First child’s id of current TextualNode

**getReffs** (*level: int = 1, subreference: MyCapytain.common.reference.\_base.BaseReference = None*)

→ `MyCapytain.common.reference._base.BaseReferenceSet`

CtsReference available at a given level

**Parameters**

- **level** – Depth required. If not set, should retrieve first encountered level (1 based)
- **subreference** – Subreference (optional)

**Returns** List of levels

**getTextualNode** (*subreference: MyCapytain.common.reference.\_base.BaseReference*) → MyCapytain.resources.prototypes.text.TextualGraph  
Retrieve a passage and store it in the object

**Parameters** **subreference** – CtsReference of the passage to retrieve

**Returns** Object representing the passage

**Raises** *TypeError* when reference is not a list or a CtsReference

**get\_creator** (*lang: str = None*) → rdflib.term.Literal  
Get the DC Creator literal value

**Parameters** **lang** – Language to retrieve

**Returns** Creator string representation

**Return type** Literal

**get\_description** (*lang: str = None*) → rdflib.term.Literal  
Get the description of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Description string representation

**Return type** Literal

**get\_subject** (*lang=None*) → rdflib.term.Literal  
Get the subject of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Subject string representation

**Return type** Literal

**get\_title** (*lang: str = None*) → rdflib.term.Literal  
Get the title of the object

**Parameters** **lang** – Lang to retrieve

**Returns** Title string representation

**Return type** Literal

**graph**

**id**

Identifier of the text

**Returns** Identifier of the text

**last**

Last CapitainsCtsPassage

**lastId**

Last child's id of current TextualNode

**metadata**

Metadata information about the text

**Returns** Collection object with metadata about the text

**Return type** Metadata

**next**

Get Next TextualNode

**nextId**

Next Id

**parent**

Parent TextualNode

**parentId**

Parent Id

**prev**

Get Previous TextualNode

**prevId**

Previous Id (Sibling)

**set\_creator** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)

Set the DC Creator literal value

**Parameters**

- **value** – Value of the creator node
- **lang** – Language in which the value is

**set\_description** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)

Set the DC Description literal value

**Parameters**

- **value** – Value of the title node
- **lang** – Language in which the value is

**set\_subject** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)

Set the DC Subject literal value

**Parameters**

- **value** – Value of the subject node
- **lang** – Language in which the value is

**set\_title** (*value: Union[rdflib.term.Literal, rdflib.term.Identifier, str], lang: str = None*)

Set the DC Title literal value

**Parameters**

- **value** – Value of the title node
- **lang** – Language in which the value is

**siblingsId**

Siblings Id

**text**

String representation of the text

**Returns** String representation of the text

## 3.6 Benchmarks

In the recent attempt to boost our system, we had a look on the performance of MyCapytain with different parser. Even if as 1.0.1 `xmlparser()` is the recommended tool, we highly recommend to switch to `lxml.objectify.parse()` parser for performance. In the following benchmark run with `timeit.sh` on the main repo (You need `PerseusDL/canonical-latinLit` somewhere ), the first line is run with `lxml.etree`, the second with `objectify` and the third with a pickled object.

### Testing on Seneca, Single Simple Passage

- 100 loops, best of 3: 4.45 msec per loop
- 100 loops, best of 3: 4.15 msec per loop
- 100 loops, best of 3: 3.75 msec per loop

### Testing range

- 100 loops, best of 3: 7.63 msec per loop
- 100 loops, best of 3: 7.72 msec per loop
- 100 loops, best of 3: 6.66 msec per loop

### Testing with a deeper architecture

- 100 loops, best of 3: 18.2 msec per loop
- 100 loops, best of 3: 14.3 msec per loop
- 100 loops, best of 3: 9.31 msec per loop

### Testing with a deeper architecture at the end

- 100 loops, best of 3: 18.2 msec per loop
- 100 loops, best of 3: 14.2 msec per loop
- 100 loops, best of 3: 9.34 msec per loop

### Testing with a deeper architecture with range

- 100 loops, best of 3: 19.3 msec per loop
- 100 loops, best of 3: 14.3 msec per loop
- 100 loops, best of 3: 9.9 msec per loop

### Testing with complicated XPATH

- 100 loops, best of 3: 751 usec per loop
- 100 loops, best of 3: 770 usec per loop
- 100 loops, best of 3: 617 usec per loop

**m**

MyCapytain.common.metadata, 26  
MyCapytain.common.utils, 28  
MyCapytain.resources.proto.text, 115  
MyCapytain.resources.prototypes.cts.inventory,  
88  
MyCapytain.resources.prototypes.metadata,  
58  
MyCapytain.resources.xml, 62  
MyCapytain.retrievers.cts5, 28  
MyCapytain.retrievers.prototypes, 30





|                   |  |                    |   |
|-------------------|--|--------------------|---|
| DEFAULT_EXPORT    | (MyCapy-<br>tain.resources.texts.local.capitains.cts.CapitainsCtsText<br>attribute), 39    | EXPORT_TO          | (MyCapy-<br>tain.resources.texts.local.capitains.cts.CapitainsCtsText<br>attribute), 39 |
| DEFAULT_EXPORT    | (MyCapy-<br>tain.resources.texts.remote.cts.CtsPassage<br>attribute), 53                   | EXPORT_TO          | (MyCapy-<br>tain.resources.texts.remote.cts.CtsPassage<br>attribute), 53                |
| DEFAULT_EXPORT    | (MyCapy-<br>tain.resources.texts.remote.cts.CtsText<br>attribute), 49                      | EXPORT_TO          | (MyCapy-<br>tain.resources.texts.remote.cts.CtsText<br>attribute), 49                   |
| DEFAULT_LANG      | (MyCapy-<br>tain.resources.texts.remote.cts.CtsText<br>attribute), 49                      | <b>F</b>           |   |
| depth             | (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage<br>attribute), 44      | first              | (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage<br>attribute), 45   |
| depth             | (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText<br>attribute), 39         | first              | (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText<br>attribute), 39      |
| depth             | (MyCapytain.resources.texts.remote.cts.CtsPassage<br>attribute), 54                        | first              | (MyCapytain.resources.texts.remote.cts.CtsPassage<br>attribute), 54                     |
| depth             | (MyCapytain.resources.texts.remote.cts.CtsText<br>attribute), 49                           | first              | (MyCapytain.resources.texts.remote.cts.CtsText<br>attribute), 49                        |
| dispatch()        | (MyCapy-<br>tain.resolvers.utils.CollectionDispatcher<br>method), 15                       | firstId            | (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPas<br>attribute), 45       |
| DtsCitation       | (class in MyCapy-<br>tain.common.reference), 26  | firstId            | (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText<br>attribute), 39      |
| DtsCitationSet    | (class in MyCapy-<br>tain.common.reference), 26  | firstId            | (MyCapytain.resources.texts.remote.cts.CtsPassage<br>attribute), 54                     |
| <b>E</b>          |  | firstId            | (MyCapytain.resources.texts.remote.cts.CtsText<br>attribute), 49                        |
| endpoint          | (MyCapytain.resolvers.cts.api.HttpCtsResolver<br>attribute), 33                            | firstUrn()         | (MyCapy-<br>tain.resources.texts.remote.cts.CtsPassage<br>static method), 54            |
| export()          | (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage<br>method), 44         | firstUrn()         | (MyCapy-<br>tain.resources.texts.remote.cts.CtsText<br>static<br>method), 49            |
| export()          | (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText<br>method), 39            | <b>G</b>           |   |
| export()          | (MyCapytain.resources.texts.remote.cts.CtsPassage<br>method), 54                           | get_creator()      | (MyCapy-<br>tain.resources.texts.local.capitains.cts.CapitainsCtsPassage<br>method), 45 |
| export()          | (MyCapytain.resources.texts.remote.cts.CtsText<br>method), 49                              | get_creator()      | (MyCapy-<br>tain.resources.texts.local.capitains.cts.CapitainsCtsText<br>method), 40    |
| export_capacities | (MyCapy-<br>tain.resources.texts.local.capitains.cts.CapitainsCtsPassage<br>attribute), 45 | get_creator()      | (MyCapy-<br>tain.resources.texts.remote.cts.CtsPassage<br>method), 55                   |
| export_capacities | (MyCapy-<br>tain.resources.texts.local.capitains.cts.CapitainsCtsText<br>attribute), 39    | get_creator()      | (MyCapy-<br>tain.resources.texts.remote.cts.CtsText<br>method),<br>51                   |
| export_capacities | (MyCapy-<br>tain.resources.texts.remote.cts.CtsPassage<br>attribute), 54                   | get_cts_metadata() | (MyCapy-<br>tain.resources.texts.local.capitains.cts.CapitainsCtsPassage<br>method), 46 |
| export_capacities | (MyCapy-<br>tain.resources.texts.remote.cts.CtsText<br>attribute), 49                      | get_cts_metadata() | (MyCapy-<br>tain.resources.texts.local.capitains.cts.CapitainsCtsText<br>method), 40    |
| EXPORT_TO         | (MyCapy-<br>tain.resources.texts.local.capitains.cts.CapitainsCtsPassage<br>attribute), 44 |                    |   |

|                                 |  |                               |  |
|---------------------------------|--|-------------------------------|--|
| <code>get_cts_metadata()</code> | (MyCapy-<br><i>tain.resources.texts.remote.cts.CtsPassage</i><br>method), 56         | <code>getLabel()</code>       | (MyCapy-<br><i>tain.resources.texts.remote.cts.CtsPassage</i><br>method), 54         |
| <code>get_cts_metadata()</code> | (MyCapy-<br><i>tain.resources.texts.remote.cts.CtsText</i><br>method), 51            | <code>getLabel()</code>       | (MyCapy-<br><i>tain.resources.texts.remote.cts.CtsText</i><br>method), 50            |
| <code>get_description()</code>  | (MyCapy-<br><i>tain.resources.texts.local.capitains.cts.Capitains</i><br>method), 46 | <code>getMetadata()</code>    | (MyCapy-<br><i>tain.resolvers.cts.api.HttpCtsResolver</i><br>method), 33             |
| <code>get_description()</code>  | (MyCapy-<br><i>tain.resources.texts.local.capitains.cts.Capitains</i><br>method), 40 | <code>getMetadata()</code>    | (MyCapy-<br><i>tain.resolvers.cts.local.CtsCapitainsLocalResolver</i><br>method), 35 |
| <code>get_description()</code>  | (MyCapy-<br><i>tain.resources.texts.remote.cts.CtsPassage</i><br>method), 56         | <code>getMetadata()</code>    | (MyCapy-<br><i>tain.resolvers.prototypes.Resolver</i><br>method), 18                 |
| <code>get_description()</code>  | (MyCapy-<br><i>tain.resources.texts.remote.cts.CtsText</i><br>method), 51            | <code>getPassagePlus()</code> | (MyCapy-<br><i>tain.resources.texts.remote.cts.CtsPassage</i><br>method), 55         |
| <code>get_subject()</code>      | (MyCapy-<br><i>tain.resources.texts.local.capitains.cts.Capitains</i><br>method), 46 | <code>getPassagePlus()</code> | (MyCapy-<br><i>tain.resources.texts.remote.cts.CtsText</i><br>method), 50            |
| <code>get_subject()</code>      | (MyCapy-<br><i>tain.resources.texts.local.capitains.cts.Capitains</i><br>method), 41 | <code>getPrevNextUrn()</code> | (MyCapy-<br><i>tain.resources.texts.remote.cts.CtsPassage</i><br>method), 55         |
| <code>get_subject()</code>      | (MyCapy-<br><i>tain.resources.texts.remote.cts.CtsPassage</i><br>method), 56         | <code>getPrevNextUrn()</code> | (MyCapy-<br><i>tain.resources.texts.remote.cts.CtsText</i><br>method), 50            |
| <code>get_subject()</code>      | (MyCapy-<br><i>tain.resources.texts.remote.cts.CtsText</i><br>method), 51            | <code>getReffs()</code>       | (MyCapy-<br><i>tain.resolvers.cts.api.HttpCtsResolver</i><br>method), 34             |
| <code>get_title()</code>        | (MyCapy-<br><i>tain.resources.texts.local.capitains.cts.Capitains</i><br>method), 46 | <code>getReffs()</code>       | (MyCapy-<br><i>tain.resolvers.cts.local.CtsCapitainsLocalResolver</i><br>method), 35 |
| <code>get_title()</code>        | (MyCapy-<br><i>tain.resources.texts.local.capitains.cts.Capitains</i><br>method), 41 | <code>getReffs()</code>       | (MyCapy-<br><i>tain.resolvers.prototypes.Resolver</i><br>method), 18                 |
| <code>get_title()</code>        | (MyCapy-<br><i>tain.resources.texts.remote.cts.CtsPassage</i><br>method), 56         | <code>getReffs()</code>       | (MyCapy-<br><i>tain.resources.texts.local.capitains.cts.Capitains</i><br>method), 45 |
| <code>get_title()</code>        | (MyCapy-<br><i>tain.resources.texts.remote.cts.CtsText</i><br>method), 51            | <code>getReffs()</code>       | (MyCapy-<br><i>tain.resources.texts.local.capitains.cts.Capitains</i><br>method), 40 |
| <code>getFirstUrn()</code>      | (MyCapy-<br><i>tain.resources.texts.remote.cts.CtsPassage</i><br>method), 54         | <code>getReffs()</code>       | (MyCapy-<br><i>tain.resources.texts.remote.cts.CtsPassage</i><br>method), 55         |
| <code>getFirstUrn()</code>      | (MyCapy-<br><i>tain.resources.texts.remote.cts.CtsText</i><br>method), 50            | <code>getReffs()</code>       | (MyCapy-<br><i>tain.resources.texts.remote.cts.CtsText</i><br>method), 50            |
| <code>getLabel()</code>         | (MyCapy-<br><i>tain.resources.texts.local.capitains.cts.Capitains</i><br>method), 45 | <code>getSiblings()</code>    | (MyCapy-<br><i>tain.resolvers.cts.api.HttpCtsResolver</i><br>method), 34             |
| <code>getLabel()</code>         | (MyCapy-<br><i>tain.resources.texts.local.capitains.cts.Capitains</i><br>method), 39 | <code>getSiblings()</code>    | (MyCapy-<br><i>tain.resolvers.cts.local.CtsCapitainsLocalResolver</i><br>method), 35 |

[getSiblings\(\)](#) (*MyCapytain.resolvers.prototypes.Resolver* method), 19  
[getTextualNode\(\)](#) (*MyCapytain.resolvers.cts.api.HttpCtsResolver* method), 34  
[getTextualNode\(\)](#) (*MyCapytain.resolvers.cts.local.CtsCapitainsLocalResolver* method), 35  
[getTextualNode\(\)](#) (*MyCapytain.resolvers.prototypes.Resolver* method), 19  
[getTextualNode\(\)](#) (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage* method), 45  
[getTextualNode\(\)](#) (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText* method), 40  
[getTextualNode\(\)](#) (*MyCapytain.resources.texts.remote.cts.CtsPassage* method), 55  
[getTextualNode\(\)](#) (*MyCapytain.resources.texts.remote.cts.CtsText* method), 50  
[getValidReff\(\)](#) (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage* method), 45  
[getValidReff\(\)](#) (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText* method), 40  
[getValidReff\(\)](#) (*MyCapytain.resources.texts.remote.cts.CtsPassage* method), 55  
[getValidReff\(\)](#) (*MyCapytain.resources.texts.remote.cts.CtsText* method), 50  
[graph](#) (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage* attribute), 46  
[graph](#) (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText* attribute), 41  
[graph](#) (*MyCapytain.resources.texts.remote.cts.CtsPassage* attribute), 56  
[graph](#) (*MyCapytain.resources.texts.remote.cts.CtsText* attribute), 51

## H

[HttpCtsResolver](#) (class in *MyCapytain.resolvers.cts.api*), 33

## I

[id](#) (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage* attribute), 46  
[id](#) (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText* attribute), 41

## L

[last](#) (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage* attribute), 46  
[last](#) (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText* attribute), 41  
[last](#) (*MyCapytain.resources.texts.remote.cts.CtsPassage* attribute), 56  
[last](#) (*MyCapytain.resources.texts.remote.cts.CtsText* attribute), 51  
[lastId](#) (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage* attribute), 46  
[lastId](#) (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText* attribute), 41  
[lastId](#) (*MyCapytain.resources.texts.remote.cts.CtsPassage* attribute), 56  
[lastId](#) (*MyCapytain.resources.texts.remote.cts.CtsText* attribute), 51

## M

[metadata](#) (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage* attribute), 46  
[metadata](#) (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText* attribute), 41  
[metadata](#) (*MyCapytain.resources.texts.remote.cts.CtsPassage* attribute), 56  
[metadata](#) (*MyCapytain.resources.texts.remote.cts.CtsText* attribute), 52  
[methods](#) (*MyCapytain.resolvers.util.CollectionDispatcher* attribute), 16  
[MyCapytain.common.metadata](#) (module), 26  
[MyCapytain.common.util](#) (module), 28  
[MyCapytain.resources.proto.text](#) (module), 115  
[MyCapytain.resources.prototype.cts.inventory](#) (module), 88  
[MyCapytain.resources.prototype.metadata](#) (module), 58  
[MyCapytain.resources.xml](#) (module), 62  
[MyCapytain.retrievers.cts5](#) (module), 28  
[MyCapytain.retrievers.prototype](#) (module), 30

## N

[next](#) (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage* attribute), 46

next (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText* attribute), 41

next (*MyCapytain.resources.texts.remote.cts.CtsPassage* attribute), 56

next (*MyCapytain.resources.texts.remote.cts.CtsText* attribute), 52

nextId (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage* attribute), 46

nextId (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText* attribute), 41

nextId (*MyCapytain.resources.texts.remote.cts.CtsPassage* attribute), 56

nextId (*MyCapytain.resources.texts.remote.cts.CtsText* attribute), 52

NodeId (class in *MyCapytain.common.reference*), 24

**P**

pagination () (*MyCapytain.resolvers.cts.local.CtsCapitainsLocalResolver* static method), 36

parent (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage* attribute), 47

parent (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText* attribute), 41

parent (*MyCapytain.resources.texts.remote.cts.CtsPassage* attribute), 57

parent (*MyCapytain.resources.texts.remote.cts.CtsText* attribute), 52

parentId (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage* attribute), 47

parentId (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText* attribute), 41

parentId (*MyCapytain.resources.texts.remote.cts.CtsPassage* attribute), 57

parentId (*MyCapytain.resources.texts.remote.cts.CtsText* attribute), 52

parse () (*MyCapytain.resolvers.cts.local.CtsCapitainsLocalResolver* method), 36

PLAINTEXT\_STRING\_JOIN (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage* attribute), 44

plaintext\_string\_join (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage* attribute), 47

PLAINTEXT\_STRING\_JOIN (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText* attribute), 39

plaintext\_string\_join (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText* attribute), 41

PLAINTEXT\_STRING\_JOIN (*MyCapytain.resources.texts.remote.cts.CtsPassage* attribute), 53

plaintext\_string\_join (*MyCapytain.resources.texts.remote.cts.CtsPassage* attribute), 57

prev (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage* attribute), 47

prev (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText* attribute), 41

prev (*MyCapytain.resources.texts.remote.cts.CtsPassage* attribute), 57

prev (*MyCapytain.resources.texts.remote.cts.CtsText* attribute), 52

prevId (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage* attribute), 47

prevId (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText* attribute), 41

prevId (*MyCapytain.resources.texts.remote.cts.CtsPassage* attribute), 57

prevId (*MyCapytain.resources.texts.remote.cts.CtsText* attribute), 52

prevnext () (*MyCapytain.resources.texts.remote.cts.CtsPassage* static method), 57

prevnext () (*MyCapytain.resources.texts.remote.cts.CtsText* static method), 52

**R**

read () (*MyCapytain.resolvers.cts.local.CtsCapitainsLocalResolver* method), 36

reference (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage* attribute), 47

reference (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText* attribute), 41

reference (*MyCapytain.resources.texts.remote.cts.CtsPassage* attribute), 57

reffs (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText* attribute), 41

reffs (*MyCapytain.resources.texts.remote.cts.CtsText* attribute), 52

Resolver (class in *MyCapytain.resolvers.prototypes*), 18

retriever (*MyCapytain.resources.texts.remote.cts.CtsPassage* attribute), 57

retriever (*MyCapytain.resources.texts.remote.cts.CtsText* attribute), 52

**S**

set\_creator () (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage* method), 47

set\_creator () (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText* method), 41

method), 42

set\_creator() (MyCapytain.resources.texts.remote.cts.CtsPassagemethod), 57

set\_creator() (MyCapytain.resources.texts.remote.cts.CtsTextmethod), 52

set\_description() (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassagemethod), 47

set\_description() (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsTextmethod), 42

set\_description() (MyCapytain.resources.texts.remote.cts.CtsPassagemethod), 57

set\_description() (MyCapytain.resources.texts.remote.cts.CtsTextmethod), 52

set\_metadata\_from\_collection() (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassagemethod), 47

set\_metadata\_from\_collection() (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsTextmethod), 42

set\_metadata\_from\_collection() (MyCapytain.resources.texts.remote.cts.CtsPassagemethod), 57

set\_metadata\_from\_collection() (MyCapytain.resources.texts.remote.cts.CtsTextmethod), 53

set\_subject() (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassagemethod), 47

set\_subject() (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsTextmethod), 42

set\_subject() (MyCapytain.resources.texts.remote.cts.CtsPassagemethod), 58

set\_subject() (MyCapytain.resources.texts.remote.cts.CtsTextmethod), 53

set\_title() (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassagemethod), 47

set\_title() (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsTextmethod), 42

set\_title() (MyCapytain.resources.texts.remote.cts.CtsPassagemethod), 58

set\_title() (MyCapytain.resources.texts.remote.cts.CtsTextmethod), 53

53

siblingsId (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassagemethod), 48

siblingsId (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsTextmethod), 42

siblingsId (MyCapytain.resources.texts.remote.cts.CtsPassagemethod), 58

siblingsId (MyCapytain.resources.texts.remote.cts.CtsTextmethod), 53

**T**

test() (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsTextmethod), 42

text (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassagemethod), 48

text (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsTextmethod), 42

text (MyCapytain.resources.texts.remote.cts.CtsPassagemethod), 58

text (MyCapytain.resources.texts.remote.cts.CtsTextmethod), 53

textObject (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassagemethod), 48

textObject (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsTextmethod), 42

toString() (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassagemethod), 48

toString() (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsTextmethod), 42

**U**

URN (class in MyCapytain.common.reference), 25

urn (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassagemethod), 48

urn (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsTextmethod), 43

urn (MyCapytain.resources.texts.remote.cts.CtsPassagemethod), 58

urn (MyCapytain.resources.texts.remote.cts.CtsTextmethod), 53

**X**

xml (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassagemethod), 48

xml (MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsTextmethod), 43

xml (*MyCapytain.resources.texts.remote.cts.CtsPassage*  
attribute), 58

xmlparse() (*MyCapytain.resolvers.cts.local.CtsCapitainsLocalResolver*  
method), 36

xpath() (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsPassage*  
method), 48

xpath() (*MyCapytain.resources.texts.local.capitains.cts.CapitainsCtsText*  
method), 43