Outline

1. Documentation Elements
2. TEI ODD
3. Roma
Elements defined by the tagdocs module: altIdent att attDef attList attRef classSpec classes code content datatype defaultVal eg egXML elementSpec equiv exemplum gi ident listRef macroSpec memberOf moduleRef moduleSpec remarks schemaSpec specDesc specGrp specGrpRef specList stringVal tag val valDesc valItem valList
The 'Documentation Elements' chapter defines elements useful in describing XML, attributes, elements, and schemas.

The TEI uses these elements in creation of the TEI Guidelines in its ODD (One Document Does-it-all) format.

The documentation elements should be used any time one is describing XML elements.

The ODD format, a TEI document using these elements in a particular way, can be used to customize the TEI to suit your needs.
The 'Documentation Elements' chapter defines elements useful in describing XML, attributes, elements, and schemas.

The TEI uses these elements in creation of the TEI Guidelines in its ODD (One Document Does-it-all) format.

The documentation elements should be used any time one is describing XML elements.

The ODD format, a TEI document using these elements in a particular way, can be used to customize the TEI to suit your needs.
Documenting XML Markup Schemes

- The ’Documentation Elements’ chapter defines elements useful in describing XML, attributes, elements, and schemas.
- The TEI uses these elements in creation of the TEI Guidelines in its ODD (One Document Does-it-all) format.
- The documentation elements should be used any time one is describing XML elements.
- The ODD format, a TEI document using these elements in a particular way, can be used to customize the TEI to suit your needs.
The ’Documentation Elements’ chapter defines elements useful in describing XML, attributes, elements, and schemas.

The TEI uses these elements in creation of the TEI Guidelines in its ODD (One Document Does-it-all) format.

The documentation elements should be used any time one is describing XML elements.

The ODD format, a TEI document using these elements in a particular way, can be used to customize the TEI to suit your needs.
Multiple Outputs

An ODD Processor (something which reads a TEI ODD file and from that creates a number of outputs), may be used to produce:

- formal reference documentation for elements, attributes, element classes, such as those provided in Appendix C of the TEI Guidelines
- detailed descriptive prose documentation, embedding some parts of the formal reference documentation, such as the tag description lists provided in this and other chapters of the Guidelines
- declarative code for one or more XML schema languages, specifically RELAX NG or W3C Schema
- declarative code for fragments which can be assembled to make up an XML Document Type Declaration.
A Multiple Outputs

An ODD Processor (something which reads a TEI ODD file and from that creates a number of outputs), may be used to produce:

- formal reference documentation for elements, attributes, element classes, such as those provided in Appendix C of the TEI Guidelines
- detailed descriptive prose documentation, embedding some parts of the formal reference documentation, such as the tag description lists provided in this and other chapters of the Guidelines
- declarative code for one or more XML schema languages, specifically RELAX NG or W3C Schema
- declarative code for fragments which can be assembled to make up an XML Document Type Declaration.
An ODD Processor (something which reads a TEI ODD file and from that creates a number of outputs), may be used to produce:

- formal reference documentation for elements, attributes, element classes, such as those provided in Appendix C of the TEI Guidelines
- detailed descriptive prose documentation, embedding some parts of the formal reference documentation, such as the tag description lists provided in this and other chapters of the Guidelines
- declarative code for one or more XML schema languages, specifically RELAX NG or W3C Schema
- declarative code for fragments which can be assembled to make up an XML Document Type Declaration.
Multiple Outputs

An ODD Processor (something which reads a TEI ODD file and from that creates a number of outputs), may be used to produce:

- formal reference documentation for elements, attributes, element classes, such as those provided in Appendix C of the TEI Guidelines
- detailed descriptive prose documentation, embedding some parts of the formal reference documentation, such as the tag description lists provided in this and other chapters of the Guidelines
- declarative code for one or more XML schema languages, specifically RELAX NG or W3C Schema
- declarative code for fragments which can be assembled to make up an XML Document Type Declaration.
Following the TEI spirit

Using the TEI means:

- Sharing a common text encoding culture
- Sharing the same vocabulary (when applicable)
- Allowing user autonomy in defining modifications (extensions, customization), but sharing the mechanisms to do so

The TEI gives you a lot of help in following these rules.
Phrase Level Documentation Elements

- `<code>` (literal code from some formal language)
- `<ident>` (an identifier for an object of some kind in a formal language)
- `<att>` (the name of an attribute appearing within running text)
- `<val>` (a single attribute value)
- `<gi>` (the name (generic identifier) of an element.)
- `<tag>` (text of a complete start- or end-tag, possibly including attribute specifications, but excluding the opening and closing markup delimiter characters)
- `<specList>` (marks where a list of descriptions is to be inserted into the prose documentation)
- `<specDesc/>` (a description of the specified element or class should be included at this point)
Phrase Level Documentation Elements

- `<code>` (literal code from some formal language)
- `<ident>` (an identifier for an object of some kind in a formal language)
- `<att>` (the name of an attribute appearing within running text)
- `<val>` (a single attribute value)
- `<gi>` (the name (generic identifier) of an element.)
- `<tag>` (text of a complete start- or end-tag, possibly including attribute specifications, but excluding the opening and closing markup delimiter characters)
- `<specList>` (marks where a list of descriptions is to be inserted into the prose documentation)
- `<specDesc/>` (a description of the specified element or class should be included at this point)
Phrase Level Documentation Elements

- `<code>` (literal code from some formal language)
- `<ident>` (an identifier for an object of some kind in a formal language)
- `<att>` (the name of an attribute appearing within running text)
- `<val>` (a single attribute value)
- `<gi>` (the name (generic identifier) of an element.)
- `<tag>` (text of a complete start- or end-tag, possibly including attribute specifications, but excluding the opening and closing markup delimiter characters)
- `<specList>` (marks where a list of descriptions is to be inserted into the prose documentation)
- `<specDesc/>` (a description of the specified element or class should be included at this point)
Phrase Level Documentation Elements

- `<code>` (literal code from some formal language)
- `<ident>` (an identifier for an object of some kind in a formal language)
- `<att>` (the name of an attribute appearing within running text)
- `<val>` (a single attribute value)
- `<gi>` (the name (generic identifier) of an element.)
- `<tag>` (text of a complete start- or end-tag, possibly including attribute specifications, but excluding the opening and closing markup delimiter characters)
- `<specList>` (marks where a list of descriptions is to be inserted into the prose documentation)
- `<specDesc/>` (a description of the specified element or class should be included at this point)
Phrase Level Documentation Elements

- `<code>` (literal code from some formal language)
- `<ident>` (an identifier for an object of some kind in a formal language)
- `<att>` (the name of an attribute appearing within running text)
- `<val>` (a single attribute value)
- `<gi>` (the name (generic identifier) of an element.)
- `<tag>` (text of a complete start- or end-tag, possibly including attribute specifications, but excluding the opening and closing markup delimiter characters)
- `<specList>` (marks where a list of descriptions is to be inserted into the prose documentation)
- `<specDesc/>` (a description of the specified element or class should be included at this point)
Phrase Level Documentation Elements

- `<code>` (literal code from some formal language)
- `<ident>` (an identifier for an object of some kind in a formal language)
- `<att>` (the name of an attribute appearing within running text)
- `<val>` (a single attribute value)
- `<gi>` (the name (generic identifier) of an element.)
- `<tag>` (text of a complete start- or end-tag, possibly including attribute specifications, but excluding the opening and closing markup delimiter characters)
- `<specList>` (marks where a list of descriptions is to be inserted into the prose documentation)
- `<specDesc/>` (a description of the specified element or class should be included at this point)
Phrase Level Documentation Elements

- `<code>` (literal code from some formal language)
- `<ident>` (an identifier for an object of some kind in a formal language)
- `<att>` (the name of an attribute appearing within running text)
- `<val>` (a single attribute value)
- `<gi>` (the name (generic identifier) of an element.)
- `<tag>` (text of a complete start- or end-tag, possibly including attribute specifications, but excluding the opening and closing markup delimiter characters)
- `<specList>` (marks where a list of descriptions is to be inserted into the prose documentation)
- `<specDesc/>` (a description of the specified element or class should be included at this point)
Phrase Level Documentation Elements

- `<code>` (literal code from some formal language)
- `<ident>` (an identifier for an object of some kind in a formal language)
- `<att>` (the name of an attribute appearing within running text)
- `<val>` (a single attribute value)
- `<gi>` (the name (generic identifier) of an element.)
- `<tag>` (text of a complete start- or end-tag, possibly including attribute specifications, but excluding the opening and closing markup delimiter characters)
- `<specList>` (marks where a list of descriptions is to be inserted into the prose documentation)
- `<specDesc/>` (a description of the specified element or class should be included at this point)
Specifying Elements

- `<elementSpec>` (documents the structure, content, and purpose of a single element type)
- `<classSpec>` (reference information for an element class)
- `<macroSpec>` (documents the function and implementation of a pattern)
<elementSpec> (documents the structure, content, and purpose of a single element type)

<classSpec> (reference information for an element class)

<macroSpec> (documents the function and implementation of a pattern)
Specification Elements

- `<elementSpec>` (documents the structure, content, and purpose of a single element type)
- `<classSpec>` (reference information for an element class)
- `<macroSpec>` (documents the function and implementation of a pattern)
Common Elements (1)

- **Description:**
  - `<remarks>` (any commentary or discussion about the usage of an element, attribute, or class)
  - `<listRef>` (a list of significant references to places where this element is discussed)

- **Examples**
  - `<exemplum>` (a single example demonstrating the use of an element)
  - `<eg>` (any kind of illustrative example)
  - `<egXML>` (a single well-formed XML example demonstrating the use of some XML element or attribute)

- **Classification**
  - `<classes>` (the classes of which the element or class is a member)
  - `<memberOf>` (class membership of the parent element or class)
Common Elements (1)

- **Description:**
  - `<remarks>` (any commentary or discussion about the usage of an element, attribute, or class)
  - `<listRef>` (a list of significant references to places where this element is discussed)

- **Examples**
  - `<exemplum>` (a single example demonstrating the use of an element)
  - `<eg>` (any kind of illustrative example)
  - `<egXML>` (a single well-formed XML example demonstrating the use of some XML element or attribute)

- **Classification**
  - `<classes>` (the classes of which the element or class is a member)
  - `<memberOf>` (class membership of the parent element or class)
Common Elements (1)

- **Description:**
  - `<remarks>` (any commentary or discussion about the usage of an element, attribute, or class)
  - `<listRef>` (a list of significant references to places where this element is discussed)

- **Examples**
  - `<exemplum>` (a single example demonstrating the use of an element)
  - `<eg>` (any kind of illustrative example)
  - `<egXML>` (a single well-formed XML example demonstrating the use of some XML element or attribute)

- **Classification**
  - `<classes>` (the classes of which the element or class is a member)
  - `<memberOf>` (class membership of the parent element or class)
Common Elements (1)

• Description:
  • <remarks> (any commentary or discussion about the usage of an element, attribute, or class)
  • <listRef> (a list of significant references to places where this element is discussed)

• Examples
  • <exemplum> (a single example demonstrating the use of an element)
  • <eg> (any kind of illustrative example)
  • <egXML> (a single well-formed XML example demonstrating the use of some XML element or attribute)

• Classification
  • <classes> (the classes of which the element or class is a member)
  • <memberOf> (class membership of the parent element or class)
Common Elements (1)

- **Description:**
  - `<remarks>` (any commentary or discussion about the usage of an element, attribute, or class)
  - `<listRef>` (a list of significant references to places where this element is discussed)

- **Examples**
  - `<exemplum>` (a single example demonstrating the use of an element)
  - `<eg>` (any kind of illustrative example)
  - `<egXML>` (a single well-formed XML example demonstrating the use of some XML element or attribute)

- **Classification**
  - `<classes>` (the classes of which the element or class is a member)
  - `<memberOf>` (class membership of the parent element or class)
Common Elements (1)

- Description:
  - `<remarks>` (any commentary or discussion about the usage of an element, attribute, or class)
  - `<listRef>` (a list of significant references to places where this element is discussed)

- Examples
  - `<exemplum>` (a single example demonstrating the use of an element)
  - `<eg>` (any kind of illustrative example)
  - `<egXML>` (a single well-formed XML example demonstrating the use of some XML element or attribute)

- Classification
  - `<classes>` (the classes of which the element or class is a member)
  - `<memberOf>` (class membership of the parent element or class)
Common Elements (1)

- **Description:**
  - `<remarks>` (any commentary or discussion about the usage of an element, attribute, or class)
  - `<listRef>` (a list of significant references to places where this element is discussed)

- **Examples**
  - `<exemplum>` (a single example demonstrating the use of an element)
  - `<eg>` (any kind of illustrative example)
  - `<egXML>` (a single well-formed XML example demonstrating the use of some XML element or attribute)

- **Classification**
  - `<classes>` (the classes of which the element or class is a member)
  - `<memberOf>` (class membership of the parent element or class)
Common Elements (2)

- **Element Specifications**
  - `<content>` (the text of a content model for the schema)
  - `<attList>` (documentation for all the attributes associated with this element, as a series of `<attDef>` elements)

- **Attributes**
  - `<attDef>` (definition of a single attribute)
  - `<datatype>` (schema datatype for the attribute value)
  - `<defaultValue>` (default declared attribute value)
  - `<valDesc>` (description of any attribute value)
  - `<valList>` (a list of attribute value items)
  - `<valItem>` (a single attribute value item)
Common Elements (2)

- **Element Specifications**
  - `<content>` (the text of a content model for the schema)
  - `<attList>` (documentation for all the attributes associated with this element, as a series of `<attDef>` elements)

- **Attributes**
  - `<attDef>` (definition of a single attribute)
  - `<datatype>` (schema datatype for the attribute value)
  - `<defaultValue>` (default declared attribute value)
  - `<valDesc>` (description of any attribute value)
  - `<valList>` (a list of attribute value items)
  - `<valItem>` (a single attribute value item)
Common Elements (2)

- **Element Specifications**
  - `<content>` (the text of a content model for the schema)
  - `<attList>` (documentation for all the attributes associated with this element, as a series of `<attDef>` elements)

- **Attributes**
  - `<attDef>` (definition of a single attribute)
  - `<datatype>` (schema datatype for the attribute value)
  - `<defaultVal>` (default declared attribute value)
  - `<valDesc>` (description of any attribute value)
  - `<valList>` (a list of attribute value items)
  - `<valItem>` (a single attribute value item)
Common Elements (2)

- **Element Specifications**
  - `<content>` (the text of a content model for the schema)
  - `<attList>` (documentation for all the attributes associated with this element, as a series of `<attDef>` elements)

- **Attributes**
  - `<attDef>` (definition of a single attribute)
  - `<datatype>` (schema datatype for the attribute value)
  - `<defaultVal>` (default declared attribute value)
  - `<valDesc>` (description of any attribute value)
  - `<valList>` (a list of attribute value items)
  - `<valItem>` (a single attribute value item)
Common Elements (2)

- **Element Specifications**
  - `<content>` (the text of a content model for the schema)
  - `<attList>` (documentation for all the attributes associated with this element, as a series of `<attDef>` elements)

- **Attributes**
  - `<attDef>` (definition of a single attribute)
  - `<datatype>` (schema datatype for the attribute value)
  - `<defaultVal>` (default declared attribute value)
  - `<valDesc>` (description of any attribute value)
  - `<valList>` (a list of attribute value items)
  - `<valItem>` (a single attribute value item)
Element Specifications
- `<content>` (the text of a content model for the schema)
- `<attList>` (documentation for all the attributes associated with this element, as a series of `<attDef>` elements)

Attributes
- `<attDef>` (definition of a single attribute)
- `<datatype>` (schema datatype for the attribute value)
- `<defaultValue>` (default declared attribute value)
- `<valDesc>` (description of any attribute value)
- `<valList>` (a list of attribute value items)
- `<valItem>` (a single attribute value item)
Common Elements (2)

- **Element Specifications**
  - `<content>` (the text of a content model for the schema)
  - `<attList>` (documentation for all the attributes associated with this element, as a series of `<attDef>` elements)

- **Attributes**
  - `<attDef>` (definition of a single attribute)
  - `<datatype>` (schema datatype for the attribute value)
  - `<defaultVal>` (default declared attribute value)
  - `<valDesc>` (description of any attribute value)
  - `<valList>` (a list of attribute value items)
  - `<valItem>` (a single attribute value item)
Common Elements (2)

- **Element Specifications**
  - `<content>` (the text of a content model for the schema)
  - `<attList>` (documentation for all the attributes associated with this element, as a series of `<attDef>` elements)

- **Attributes**
  - `<attDef>` (definition of a single attribute)
  - `<datatype>` (schema datatype for the attribute value)
  - `<defaultVal>` (default declared attribute value)
  - `<valDesc>` (description of any attribute value)
  - `<valList>` (a list of attribute value items)
  - `<valItem>` (a single attribute value item)
The TEI uses the Documentation Elements to write the TEI Guidelines. From the Guidelines themselves, the associated schemas and element reference documentation are generated.
Important ODD concepts

The TEI's literary programming with ODD (One Document Does it all) provides:

- Schema specification
- User oriented documentation
- Modularity: all specifications pertaining to a coherent sub-domain of the TEI
- Classes: identifying shared behaviours or semantics
- Extensibility: a consequence of the above mechanisms
The TEI ODD in practice

The TEI Guidelines, its schema, and its schema fragments, are all produced from a single XML resource containing:

1. Descriptive prose (lots of it)
2. Examples of usage (plenty)
3. Formal declarations for components of the TEI Abstract Model:
   - elements and attributes
   - modules
   - classes and macros
Possibilities of customizing the TEI

The TEI has over 20 modules. A working project will:

- Choose the modules they need
- Probably narrow the set of elements within each module
- Probably add local datatype constraints
- Possibly add new elements/attributes in other namespaces
- Possibly localize the names of elements
Under the hood

TEI customizations are themselves expressed in TEI XML, using elements from the tagdocs module mentioned above. For example:

```xml
<schemaSpec ident="myTEILite">
  <desc>This is TEI Lite with simplified heads</desc>
  <moduleRef key="tei"/>
  <moduleRef key="core"/>
  <moduleRef key="textstructure"/>
  <moduleRef key="header"/>
  <moduleRef key="linking"/>
  <elementSpec ident="head" mode="change">
    <content><rng:text/></content>
  </elementSpec>
</schemaSpec>
```

produces something like TEI Lite, with a slight change
ODD processors

- The TEI maintains a library of XSLT scripts that can generate
  - The TEI Guidelines in canonical TEI XML format
  - The Guidelines in HTML or PDF
  - RELAXNG, DTD, or W3C schema fragments
- The same library is used by the customization layer to generate
  - project-specific documentation
  - project-specific schemas
  - translations into other (human) languages
- We use eXist as a database for extracting material from the P5 sources
The TEI abstract model

- The TEI abstract model sees a markup scheme (a schema) as consisting of a number of discrete modules, which can be combined more or less as required.
- A schema is made by combining references to modules and optional element over-rides or additions.
- Each element declares the module it belongs to: elements cannot appear in more than one module.
- Each module extends the range of elements and attributes available by adding new members to existing classes of elements, or by defining new classes.
Within the class system, TEI elements have to be defined using some language notation; choices include:

1. using XML DTD language (as in older versions of the TEI)
2. using W3C Schema language
3. using the RELAXNG schema language
4. inventing an entirely new abstract language for later transformation to specific schema language

We chose a combination of 3 and 4 — using our abstract language, but switching to RELAXNG for content modelling.
Expressing constraints in XML language is too attractive to forego.
There is a clamour for better datatyping than DTDs have.
The schema languages are so good, it is silly to reinvent them.
But we like our class system and literate programming.
DTDs are not XML, and need specialist software
W3C schema is not consistently implemented, its documentation is vast and confusing, and it looks over-complex
RELAXNG on the other hand...
  - uncluttered design
  - good documentation
  - multiple open source 100%-complete implementations
  - ISO standard
  - useful features for multipurpose structural validation

No contest...
An Example ODD

```xml
<elementSpec module="spoken" ident="pause">
  <classes>
    <memberOf key="model.divPart.spoken"/>
    <memberOf key="att.timed"/>
    <memberOf key="att.typed"/>
  </classes>
  <content> <rng:empty/> </content>
  <attList>
    <attDef ident="who" usage="opt">
      <gloss>A unique identifier</gloss>
      <desc>supplies the identifier of the person or group pausing.
        Its value is the identifier of a <gi>person</gi> or <gi>persGrp</gi>
        element in the TEI header.</desc>
      <datatype><rng:ref name="data.pointer"/></datatype>
    </attDef>
  </attList>
  <desc>a pause either between or within utterances.</desc>
</elementSpec>
```
 element pause  pause.content, pause.attributes
 pause.content = empty
 pause.attributes =
   att.global.attributes,
   att.timed.attributes,
   att.typed.attributes,
   att.ascribed.attributes,
 model.divPart.spoken |= pause
 att.timed  |= pause
 att.typed  |= pause
 att.ascribed  |= pause
<!ELEMENT %n.pause; %om.RR; EMPTY>
<!ATTLIST %n.pause;
  %att.global.attributes;
  %att.timed.attributes;
  %att.typed.attributes;
  %att.ascribed.attributes;>
<!ENTITY % model.divPart.spoken
  "%x.model.divPart.spoken; %n.event; | %n.kinesic;
  | %n.pause; | %n.shift; | %n.u;
Using and Customizing the TEI: Documentation Elements, Roma and ODD

Or documentation

**<pause/>**

**<pause/>** a pause either between or within utterances. [8.3.2 Pausing](#)

<table>
<thead>
<tr>
<th>Module</th>
<th>spoken — <a href="#">8 Transcriptions of Speech</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td><a href="#">att.timed, att.typed, att.ascribed</a></td>
</tr>
</tbody>
</table>

### Declaration

```xml
element pause
{
    att.global.attributes,
    att.timed.attributes,
    att.duration.w3c.attributes,
    att.typed.attributes,
    att.ascribed.attributes,
    empty
}
```

### Example

```xml
<pause dur="PT42S" type="pregnant"/>
```

### Contained by

`model.global.spoken`

### May contain

Empty element
Overriding an attribute value-list in a TEI ODD

```xml
<elementSpec ident="list" module="core">
  <classes>
    <memberOf key="att.typed"/>
  </classes>
  <attDef ident="type" mode="replace">
    <valList type="closed">
      <valItem ident="ordered">
        <gloss>Items are ordered</gloss>
      </valItem>
      <valItem ident="bulleted">
        <gloss>Items are bulleted</gloss>
      </valItem>
      <valItem ident="frabjous">
        <gloss>Items are frabjous</gloss>
      </valItem>
    </valList>
  </attDef>
</elementSpec>
```
Modifying TEI objects

Understanding classes is critical.

- They group together elements with the same role in the TEI architecture
- They group together elements with the same syntactic behaviour
- Classes can provide attributes for groups of like-minded elements
- The elements in the class will appear in the same content models

The class defines a group of elements belonging to the same family of concepts, elements declare themselves as belonging to a class.
Uniformity of description

- modules, elements, attributes, value-lists are treated uniformly
- each has an identifier, a gloss, a description, and one or more equivalents
- each can be added, changed, replaced, deleted within a given context
- for example, membership in the \texttt{att.type} class gives you a generic type attribute, which can be over-ridden for specific class members
But the TEI knows you don’t want to necessarily have to write TEI ODDs in order to customize the TEI! So it has provided Roma, which is a command-line script, and corresponding web front-end to help you do this.

The people behind Roma are:

**Arno Mittelbach**  Initial programming

**Sebastian Rahtz**  Maintenance and frequent improvements

**Ioan Bernevig**  A ’Sanity Checker’ addition

But it is available from TEI’s Sourceforge SVN Repository, and I’m sure Sebastian would be happy for others to provide further enhancements.
Imagine that you have seen your colleague next door doing some encoding with the TEI and want to do the same thing:

- Go to Roma at http://tei.oucs.ox.ac.uk/Roma/
- Toy with the user profile [Customize]
- Generate a schema [Schema]
- Make a trial with the editor, creating a simple document
- Get back to Roma and make basic documentation
Roma: generating validators for the TEI

These pages will help you design your own TEI validator, as a DTD, RELAXNG or W3C Schema.

Create a new or upload existing customization

- Build schema (Create a new customisation by adding elements and modules to the smallest recommended schema)
- Reduce schema (Create a new customisation by removing elements and modules from the largest possible schema)
- Create customization from template
- Open existing customization

Submit

Search TEI Guide
### Roma: Customizing the TEI: Documentation Elements, Roma and ODD

#### James Cummings

Documentation Elements

#### TEI ODD

Roma

---

**Roma: generating validators for the TEI**

**Set your parameters**

<table>
<thead>
<tr>
<th>Title</th>
<th>My TEI Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename</td>
<td>myTei</td>
</tr>
<tr>
<td>Prefix for TEI</td>
<td></td>
</tr>
<tr>
<td>pattern names</td>
<td></td>
</tr>
<tr>
<td>in schema</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>English</td>
</tr>
<tr>
<td></td>
<td>Deutsch</td>
</tr>
<tr>
<td></td>
<td>Italiano</td>
</tr>
<tr>
<td></td>
<td>Español</td>
</tr>
<tr>
<td></td>
<td>Français</td>
</tr>
<tr>
<td></td>
<td>Portugues</td>
</tr>
<tr>
<td></td>
<td>Russian</td>
</tr>
<tr>
<td></td>
<td>Svenska</td>
</tr>
<tr>
<td></td>
<td>日本語</td>
</tr>
<tr>
<td></td>
<td>中文</td>
</tr>
</tbody>
</table>

**Author name**

generated by Roma 3.0

**Description**

My TEI Customization

- starts with modules `tei`, `core`, `header`, and `textstructure`

---

Submit

---

Search TEI database

---

Roma was written by Arno Mittelbach and is maintained by Sebastian Rahtz. Sanity check written by Joan Bernevig. Please direct queries to the [TEI at Oxford](https://tei-corpus.github.io) project.

This is Roma version 3.0, last updated 2007-10-21.
Roma: Schema

Time to give you a schema

Creating a schema

Which format do you prefer?

- Relax NG schema (compact syntax)
- Relax NG schema (compact syntax)
- Relax NG schema (XML syntax)
- W3C schema
- DTD

Submit

Search TEI database

Roma was written by Arno Mitselbach and is maintained by Sebastian Rahtz. Sanity check written by Ioan Bernevig. Please direct queries to the TEI@Oxford project. This is Roma version 3.0, last updated 2007-10-21.
Roma: generating validators for the TEI

Documentation?

Getting some nice documentation

Which output would you prefer?

- html
- PDF
- TEI Lite
- Tci

Submit

Search TEI database

Roma was written by Arno Mittelbach and is maintained by Sebastian Rahtz. Sanit check written by Iain Bernevig. Please direct queries to the TEI @ Oxford project. This is Roma version 3.0, last updated 2007-10-21.
Suppose you now feel you want to use some more of the TEI, but not all of it

- Go to Roma…
- Look at [Modules]
- Explore default modules by pointing to main elements (by order of interest). You can throw away most things, but
  - In textstructure, you should really keep <TEI>, <text>, <body> and <div>
  - In core, most people need <p>, <q>, <list>, <pb/> and <head>
  - From header, keep everything unless you really understand the details

- Start checking out elements
- Make editorial choices (numbered vs. unnumbered divs)
**Roma: generating validators for the TEI**

### Modules

<table>
<thead>
<tr>
<th>List of TEI Modules</th>
<th>List of selected Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Module name</strong></td>
<td><strong>remove</strong></td>
</tr>
<tr>
<td>add analysis</td>
<td>core</td>
</tr>
<tr>
<td>add certainty</td>
<td>tei</td>
</tr>
<tr>
<td>add core</td>
<td>header</td>
</tr>
<tr>
<td>add corpus</td>
<td>textstructure</td>
</tr>
<tr>
<td>add declarefs</td>
<td></td>
</tr>
<tr>
<td>add dictionaries</td>
<td></td>
</tr>
<tr>
<td>add drama</td>
<td></td>
</tr>
<tr>
<td>add figures</td>
<td></td>
</tr>
<tr>
<td>add gaij</td>
<td></td>
</tr>
<tr>
<td>add header</td>
<td></td>
</tr>
<tr>
<td>add iso-fs</td>
<td></td>
</tr>
<tr>
<td>add linking</td>
<td></td>
</tr>
<tr>
<td>add mdescription</td>
<td></td>
</tr>
<tr>
<td>add namesdtales</td>
<td></td>
</tr>
<tr>
<td>add nets</td>
<td></td>
</tr>
<tr>
<td>add spoken</td>
<td></td>
</tr>
<tr>
<td>add tagdocs</td>
<td></td>
</tr>
<tr>
<td>add textcrit</td>
<td></td>
</tr>
<tr>
<td>add textstructure</td>
<td></td>
</tr>
</tbody>
</table>
# Roma: Change Module

## Change module

The Roma change module is used to modify the TEI: Documentation Elements, Roma and ODD.

### List of elements in module: core

<table>
<thead>
<tr>
<th>Include</th>
<th>Exclude</th>
<th>Tag name</th>
<th>Description</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>abbr</td>
<td>contains an abbreviation of any sort.</td>
<td>Change attributes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>add</td>
<td>contains letters, words, or phrases inserted in the text by an author, scribe, annotator, or corrector.</td>
<td>Change attributes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>addrLine</td>
<td>contains one line of a postal address.</td>
<td>Change attributes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>address</td>
<td>contains a postal address, for example of a publisher, an organization, or an individual.</td>
<td>Change attributes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>altIdent</td>
<td>supplies the recommended XML name for an element, class, attribute, etc. in some language.</td>
<td>Change attributes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>analytic</td>
<td>contains bibliographic elements describing an item (e.g. an article or poem) published within a monograph or journal and not as an independent publication.</td>
<td>Change attributes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>author</td>
<td>in a bibliographic reference, contains the name of the author(s), personal or corporate, of a work; the primary statement of responsibility for any bibliographic item.</td>
<td>Change attributes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bibl</td>
<td>contains a loosely-structured bibliographic citation of which the sub-components may or may not be explicitly tagged.</td>
<td>Change attributes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>biblScope</td>
<td>defines the scope of a bibliographic reference, for example by year, journal, or volume.</td>
<td>Change attributes</td>
</tr>
</tbody>
</table>
# Roma: Change Attributes

## TEI: generating validators for the TEI

### Added Attributes

<table>
<thead>
<tr>
<th>Change attribute</th>
<th>Include</th>
<th>Exclude</th>
<th>Tag name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td></td>
<td></td>
<td>type</td>
<td>allows the encoder to classify the abbreviation according to some convenient typology.</td>
</tr>
<tr>
<td>xml:space</td>
<td></td>
<td></td>
<td>xml:space</td>
<td>signals an intention that white space should be preserved by applications</td>
</tr>
<tr>
<td>xml:tid</td>
<td></td>
<td></td>
<td>xml:tid</td>
<td>provides a unique identifier for the element bearing the attribute.</td>
</tr>
<tr>
<td>n</td>
<td></td>
<td></td>
<td>n</td>
<td>gives a number (or other label) for an element, which is not necessarily unique within the document.</td>
</tr>
<tr>
<td>xml:lang</td>
<td></td>
<td></td>
<td>xml:lang</td>
<td>indicates the language of the element content using a tag generated according to BCP 47</td>
</tr>
<tr>
<td>rend</td>
<td></td>
<td></td>
<td>rend</td>
<td>indicates how the element in question was rendered or presented in the source text.</td>
</tr>
<tr>
<td>rendition</td>
<td></td>
<td></td>
<td>rendition</td>
<td>points to a description of the rendering or presentation used for this element in the source text.</td>
</tr>
<tr>
<td>xml:base</td>
<td></td>
<td></td>
<td>xml:base</td>
<td>provides a base URI reference with which applications can resolve relative URI references into absolute URI references.</td>
</tr>
</tbody>
</table>
Roma: Change Attribute Values

Add some attributes

Add a new attribute

Attribute name: type
Class name:
Is it optional?
- yes
- no
Contents:
- data.enumerated
- > = 1
- <= 1
Default value:
Closed list?
- yes
- no
List of values:
- red, blue, green, purple, pink, yellow, other
Description:
Allows the encoder to classify the abbreviation according to some convenient typology.

Submit Query

Search TEI database

Roma was written by Arno Mittelbach and is maintained by Sebastian Ranz. Sanity check written by Joan Bermevig. Please direct queries to the TEI@Oxford project. This is Roma version 3.0, last updated 2007-10-21.
Roma: Change Language

Roma: generating validators for the TEI

Choose a different language

Do you want the output schema to use a different language?

- Language for element and attribute names
  - English
  - Deutsch
  - Español
  - Italiano
  - Français
  - 日本語
  - 中文

- Language for documentation
  - English
  - Deutsch
  - Español
  - Italiano
  - Français
  - 日本語
  - 中文

Submit Query

Search TEI database

Roma was written by Arno Mittelbach and is maintained by Sebastian Rahtz. Sanity check written by Joan Bernvig. Please direct queries to the TEI@Oxford project. This is Roma version 3.0, last updated 2007-10-21.
Roma: Sanity Checker

Roma: generating validators for the TEI

New  Customize  Language  Modules  Add Elements  Change Classes  Schema  Documentation  Save Customization  Sanity Checker

Progress: 100%

Schema is broken!

Warning: teiCorpus is not reachable from root

Warning: divGen is not reachable from root

In measureGrp
  text does not exist

In TEI
  text does not exist
  text does not exist
  text does not exist

Search TEI database

Roma was written by Amo Mittelbach and is maintained by Sebastian Rahtz. Sanity check written by Ioan Borovil. Please direct queries to the TEI @ Oxford project.

This is Roma version 3.0, last updated 2007-10-21.
A word of caution

Remember

- The TEI is not a monolithic environment
- Very few things are really mandatory . . .
- . . . but the TEI is more than just a market place
- Basic document structure must be preserved

The TEI is a powerful environment for working with elements and producing documentation, but do not abuse it.
Next James will lead us in an exercise in using Roma.