ACL
Cancer Research UK

Advanced Computation Laboratory,
Cancer Research UK
44 Lincoln's Inn Fields
London WC2A 3PX
Tel: 020 7269 3627 Fax: 020 7269 3186

CREDO-2005-001

Tallis Interface Primer

Rory Steele

Creation Date	14 February 2005
Last Modification	14 February 2005
Revision	8
Circulation	unrestricted
Status	Review by Tony Rose

1. What is Tallis?

TALLIS is a new suite of software tools to support authoring, publishing and enacting clinical knowledge applications over the WWW. As far as we are aware, Tallis is the only toolset currently available for publishing web-enactable guidelines. The suite of tools includes a composer to create the guidelines, an engine to enact them and an interface to display them via a web-browser.

2. Overview

Once a publet has been saved or published to a publet repository (server) it is instantly accessible over the Internet via the Tallis technology, without the need for any web development. This is because the Tallis technology incorporates a user interface (Tallis User Interface - TUI) that has the ability to dynamically generate a web page for each PRO*forma* task, as and when it becomes active during enactment. In most cases this is adequate, but often the ability to use a pre-compiled, customised page would greatly enhance the enactment and user experience. Customised pages enable more control over the layout of enactment web pages and over the way information is present to the user. They provide the ability to add additional functions and tools to complement the publet, such as images, links to further information and supporting material, widgets and even applets to aid the user during the enactment process (see **Figs. 1a** and **1b**). Using standard JSP tag libraries, it also possible to perform additional functions at the server, such as automated database queries and mail processing.

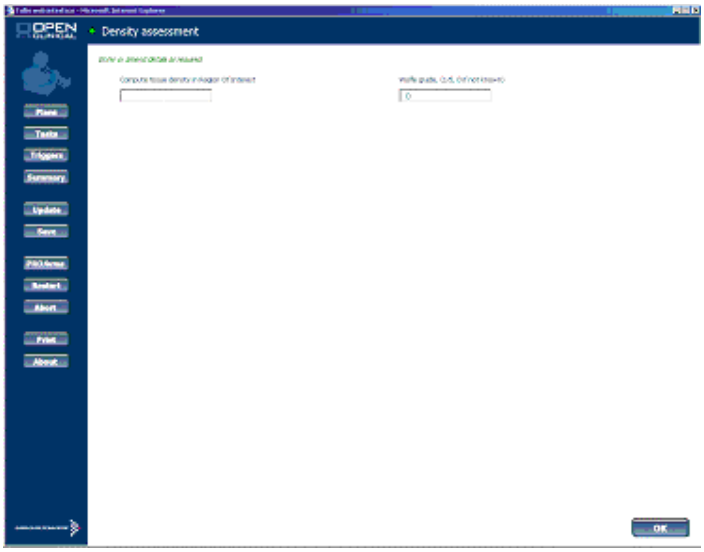

Figure 1a: A web page for an enquiry – dynamically generated by the Tallis user interface

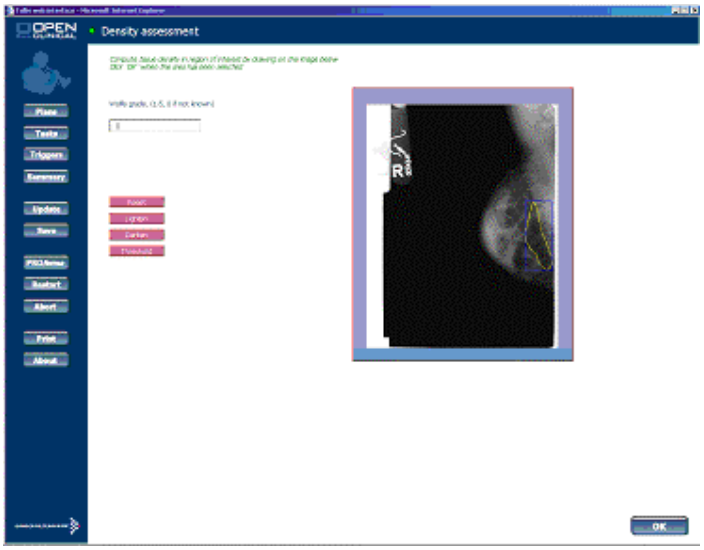

Figure 1b: An enquiry which has specified a customised page to be used by the Tallis user interface

Customised pages can be created from scratch using standard web page authoring software tools. However, the dynamically generated web pages produced by the TUI can be saved, customised and then reused the next time the publet is accessed. This method enables the TUI to be used as a rapid application development (RAD) tool and can reduce development considerably.

3. Tallis user interface

The Tallis user interface is divided into a number of sections with a specific *Cascading-style Sheet* (CSS) and *JavaServerPage* (JSP) for each section (see **Fig 2**). Associated with each layout is a JavaScript file to provide the dynamic functionality of that page. The content of each section also changes, depending on what information that page needs to convey.

For example, the "Header" section is used to provide context information for that page. In the case of viewing a task, the caption of that task is displayed. For other pages, a short description of that page's functionality is displayed.

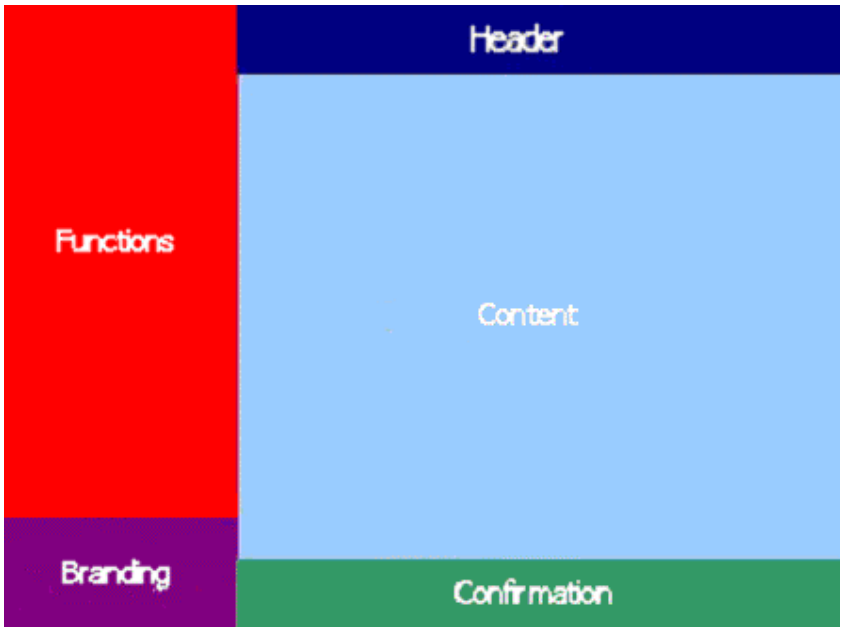

Figure 2: The default layout of the Tallis user interface

3.1 Dynamic generation of web-page content

As each task (plan, keystone, action, enquiry or decision) becomes active during enactment, the interface dynamically generates web content, depending on task type and properties. For each task, the generated content includes:

- Plans
  A graphical representation of the plan's tasks, similar to view used to author the guideline within the Tallis Composer. By default, the rendering of plans and their return to the user during enactment is disabled. The guideline may override this behaviour by the use of context properties (for further details see **Section 3.2.1.1**)
- Tasks
  Any information present within the context field of the task is rendered via the web-browser
- Actions
  Any information present within the context field of the action is rendered via the web-browser. Any information present within the procedure of the action is rendered beneath the context-provided information.
- Enquiries
  Any information present within the context field of the enquiry is rendered via the web-browser. Any information present within the sources of the enquiry is rendered beneath the context-provided information as an HTML form.
- Decisions
  Any information present within the context field of the decision is rendered via the web-browser. If the decision contains any sources, they are rendered beneath the context-provided information as an HTML form. If no sources are defined, or the sources defined have been

submitted via their HTML form, the candidates are rendered beneath the context-provided information as a group of checkboxes/radio-buttons (depending on whether the decision supports multiple candidates). The candidate can be expanded to display the relevant arguments for that candidate.

The generated content is created within the following comment tags to allow ease of location:

```
<!-- start of automatically generated content -->
        generated content
<!-- end of automatically generated content -->
```

3.2 Customisation of the Tallis user interface

It is beyond the scope of this document to describe the HTML, CSS or JavaScript standards and so it is assumed that the reader has a working knowledge of HTML, CSS and JavaScript.

3.2.1 Guideline context properties

A number of properties can also be set at the guideline level. These properties are concerned with more general processing of the guideline during enactment. The properties, their default value (in parenthesis) and a short explanation of their semantics are listed below:

- *tallis.planAware* (false). Flag to indicate whether the guideline should return plan tasks (if they have a state of *in_progress*) to the user during enactment
- *tallis.activeAbort* (true). Flag to indicate whether the guideline complete when the guideline has no more tasks in the *in_progress* state
- *tallis.content* (null). URI path for the mapping file to render customised content guideline content

3.2.1.1 tallis.planAware

If the value of the *tallis.planAware* property is set to true, plans that have a state of *in_progress* will be returned to the user for rendering. The default presentation is a graphical representation of the plan's tasks. However, the plan could specify the necessary properties to return a pre-specified page, possibly presenting the available triggers for that plan.

3.2.1.2 tallis.activeAbort

If the value of the *tallis.activeAbort* property is set to true, even when the root plan has a state of *in_progress*, if no subtasks have a state of *in_progress* the guideline will complete. However, in some circumstances this may not be the required behaviour. For example, the root plan of the guideline may contain triggerable tasks, which by their nature would remain in the *dormant* state until triggered. The author of the guideline may wish for the enactment to continue until explicitly terminated and allow the tasks to be triggered as and when they are required.

3.2.1.3 tallis.content

The value of the *tallis.content* property is a URI to a XML file conforming to the schema 'http://www.cancerresearchuk.org/acl/proforma/content#'. The schema details the options that a developer can use to customise the content delivered by the TUI. The value provided must be either an application relative URI or a valid URL. For example:

- /customisations/mappings/mapping.xml
- http://www.openclinical.org/ocnet/customisations/mappings/decision.xml

3.2.2 Guideline content mapping

3.2.2.1 Generic component properties

Certain PRO*forma* components can be mapped to alternative content via the use of a XML mapping file. The mapping between the alternative content and the enactable process is achieved by binding a definition attribute in the XML to the name of the component in the PRO*forma* content. An alternative caption and description can also be provided via sub-elements in the XML. For example:

```
…
<argument definition="largeArgument">
  <caption>
    <p style="color: red;">Large Argument In Red</p>
  </caption>
  <description>
    This contains a large description of the <b>argument</b>
  </description>
</argument>
…
```

Note the use of inline HTML content within the mapped content, removing the need for format-specific information to be contained within the original PRO*forma*.

3.2.2.2 Generic task properties

Each task component also has a generic set of properties to control rendering during enactment, including:

- *uri* (optional attribute). A path to a specific page of content to be used rather than the automatically generated content.
- *script* (optional attribute). A path to a specific JavaScript file to be included in the returned HTML page.
- *stylesheet* (optional attribute). A path to a specific CSS stylesheet to be included in the returned HTML page.
- *confirmation* (optional attribute). A javascript function name to be called when the user clicks on the "OK" button to confirm a task.

- *delegate* (optional attribute). A pointer to a definition name used elsewhere within the mapping file to which rendering should be delegated. This is used for grouping a set of tasks to one page. See **Section 3.2.2.6** for more details.

- *context* (optional sub-element). Can take any valid XML content.

3.2.2.3 Action properties

Actions extend the base task properties by allowing an optional *procedure* sub-element, which can take any valid XML content.

3.2.2.4 Decision properties

Decisions may have the following optional properties:

- *maxRecommendations* (optional attribute). An integer value specifying the maximum number of candidates that may be presented to the user as recommended, irrespective of how many are actually recommended by the enactment engine.

- *comparator* (optional attribute). A string value that resolves to a java class implementing *java.util.Comparator*. This class will be used to sort recommended candidates.

- *candidates* (optional sub-element). The candidates element can contain one-or-more candidate sub-elements, which itself can contain an optional arguments sub-element. Each arguments sub-element can contain one-or-more argument sub-elements. The candidate and argument sub-elements are used to map generic component properties (see Section 3.2.2.1).

3.2.2.5 Enquiry properties

Enquiries may have the following optional sub-elements:

- *sources* (optional). The sources sub-element can contain one-or-more source sub-elements, which has two optional attributes: position and dropdown (see Section 3.2.2.1 for the generic component properties). The position attribute can take an integer value to specify the relative position the source takes in the list of sources presented to the user. The dropdown attribute is a boolean flag to indicate whether 'set-of' values should be rendered as a drop-down list or a combo box (the default).

- *dependency* (optional). A number of dependency sub-elements can be used to control the status of dataitem form fields, listed as content within the element tag. Each element has two mandatory attributes: source and keyword. The source attribute binds to a source name within the enquiry and the keyword attribute is used to control the enabled/disabled status of the listed fields. For example:

```
…
<enquiry definition=”enquiryName”>
  <sources>
    <source definition=”sourceA” position=”1”/>
    <source definition=”sourceB” position=”3”/>
    <source definition=”sourceC” position=”2”/>
    <source definition=”sourceD” position=”4”/>
  </sources>
  <dependency source=”sourceA” keyword=”never”>
    sourceC source D
  </dependency>
…
```

In the example above, source C and sourced fields are disabled if the value of sourceA is "never". When this value is changed, the two form fields become enabled. If the value changes back to "never", the fields revert back to disabled and any values they may contain are NOT submitted to the engine during confirmation. This facility is useful for when some data relies upon values of other fields, i.e. it is senseless to ask a man when they started taking contraceptives or how many children they have given birth to.

3.2.2.6 Plan properties

Plans extend the base task properties by allowing an optional *grouping* sub-element. The grouping element must contain valid task definitions names from within the mapping file. The task elements must also use their delegate attribute to bind to the name of the plan mapping. Any tasks listed in the grouping and that are in the *in_progress* state are render within the same page rather than individually.

4. Enactable content

4.1 Rapid application development

The TUI dynamically creates web pages for tasks that don't have the mappings to specify alternative content. The generation process could therefore be used as a rapid application development (RAD) tool, as the HTML code for could be saved, edited and reused by Tallis during future enactments of the guideline.

The following steps can be followed to use the dynamically generated code produced by the TUI:

- start the enactment process
  Run through the guideline until you reach the task you wish associate with enactable content
- capture the generated code
  Right-click on the page in question, select 'view-source', copy the html between the comment tags (see above) and save the resulting text.
- edit the generated code
  HTML code editing can be performed either within Microsoft Notepad or a special purpose HTML authoring tool, such a Dreamweaver or Homesite. These sophisticated tools provide visual environments that aid code editing and enable the user to design faster and code more easily.

- save the generated code
  The new file MUST be accessible to the Tallis web-application you wish to run (see Appendix A). This does not mean that the file must be contained within the Tomcat installation running the Tallis interface, only that it can be delivered via HTTP.
- edit the PROforma guideline
  Add the necessary context properties (refer back to **Section 3.2.1** for an example of the required properties).

4.2 Required content for task specified pages

Specific information is required by the server for a task to be processed (ex. task name), with different task types requiring different information for a request to be successful. If minimal editing of the automatically generated content is performed (ex. structural changes to layout), this is generally not an issue for the content author. However, if the submission process is altered (via the *confirmation* property in the mapping file) or the content has been built from scratch, this additional information may need to be explicitly managed by the content author.

4.2.1 Actions, plans and "keystone" tasks

The TUI generates a "hidden" form for these task types, which will contain all the information required by the server for the correct processing of the task in question. No HTML form or extra content management is required.

4.2.2 Decisions

A decision is required to submit which candidate(s) the user wishes to commit for the task to be successfully processed. Care must be taken not to change the field names that are generated (HTML input and select tags).

If the decision is also authored to request specific dataitems, these fields will also need to be managed (see **Section 4.2.3** below for further details).

4.2.3 Enquiries

An enquiry is required to submit values for dataitems marked as mandatory for the task to be successfully processed. Care must be taken not to change the field names that are generated (HTML input and select tags).

5. Tallis tag libraries

It is beyond the scope of this document to describe the JSP, JSTL, Struts or Tiles standards and so it is assumed that the reader has a working knowledge of (or desire to learn) Java serverside technologies.

To assist the author wishing to develop specific content pages for a specific task, a set of JSP tag libraries have been developed to ease in their creation (see Appendix B for details). Note that the automatically generated content is itself created via a set of pages that use these tag libraries. These pages can be found within the directory *TOMCAT_HOME*/webapps/tallis/runtime/content (see

Appendix A for details) for user wishing to become more familiar with using JSPs and tag libraries.

Attributes within tags may be required (if not present an exception is thrown by Tomcat) or optional. If an attribute is required this will noted in parentheses. If an attribute is optional but provides a default value, this too will be noted in parentheses. Unless explicitly stated, all scripting variables exist in the package org.cruk.struts.shared.beans.

5.1 Tallis

The Tallis tag library provides **read-only** functionality to the set of PROforma components within the engine. The library is defined in the file *TOMCAT_HOME*/webapps/*WEB_APP*/WEB-INF/tld/tallis.tld (see Appendix A for details) and can be imported into your JSP with the URI http://acl.icnet.uk/tallis.

5.1.1 invalidate

The *invalidate* tag marks the current session as aborted. The PROforma engine is destroyed and no further processing of the guideline in question can be performed.

5.1.1.1 Attributes

None

5.1.1.2 Constraints

None

5.1.2 tallis-url

The *tallis-url* tag is used to rewrite a relative URL by appending the contextKey request parameter. This value is then stored in the scope specified by the scope attribute. The contextKey represents the ID of the PROforma engine being used to run the enactment process of the guideline in question.

5.1.2.1 Attributes

- action (**required**) - A relative URL
- var (**required**) - An ID by which to associate the URL in some context for use within script expressions
- scope (*page*) - The scope in which the URL is stores. May have values of page, request, session or application
- paramId - In conjuction with paramValue below, used to append an additional request parameter onto the action URL
- paramValue - See above
- parameters - A scripting variable of class java.util.Map used to create further request parameters for appending to the action URL

5.1.2.2 Constraints

None

5.1.3 action

The *action* tag is used to represent a PRO*forma* action and provide access to its properties.

5.1.3.1 Attributes

- name - The name of the PRO*forma* action, as specified within the PRO*forma* guideline
- action - A scripting variable of class ActionBean
- var - An ID by which to associate the ActionBean for use within script expressions (will be of *page* scope)
- style attributes (see **Section 5.3** for further details)
- script attributes (see **Section 5.4** for further details)

5.1.3.2 Constraints

Either the name or action attribute must resolve to a valid task specified within the PRO*forma* guideline.

5.1.4 decision

The *decision* tag is used to represent a PRO*forma* decision and provide access to its properties.

5.1.4.1 Attributes

- name - The name of the PRO*forma* decision, as specified within the PRO*forma* guideline
- decision - A scripting variable of class DecisionBean
- var - An ID by which to associate the DecisionBean for use within script expressions (will be of *page* scope)
- style attributes (see **Section 5.3** for further details)
- script attributes (see **Section 5.4** for further details)

5.1.4.2 Constraints

Either the name or decision attribute must resolve to a valid task specified within the PRO*forma* guideline.

5.1.5 enquiry

The *enquiry* tag is used to represent a PRO*forma* enquiry and provide access to its properties.

5.1.5.1 Attributes

- name - The name of the PRO*forma* enquiry, as specified within the PRO*forma* guideline
- enquiry - A scripting variable of class EnquiryBean
- var - An ID by which to associate the EnquiryBean for use within script expressions (will be of *page* scope)
- style attributes (see **Section 5.3** for further details)
- script attributes (see **Section 5.4** for further details)

5.1.5.2 Constraints

Either the name or enquiry attribute must resolve to a valid task specified within the PRO*forma* guideline.

5.1.6 plan

The *plan* tag is used to represent a PRO*forma* plan and provide access to its properties.

5.1.6.1 Attributes

- rootPlan (*false*) - A boolean flag used to indicate whether this tag should refer to the root plan of the PRO*forma* guideline
- name - The name of the PRO*forma* plan, as specified within the PRO*forma* guideline
- plan - A scripting variable of class PlanBean
- var - An ID by which to associate the PlanBean for use within script expressions (will be of *page* scope)
- style attributes (see **Section 5.3** for further details)
- script attributes (see **Section 5.4** for further details)

5.1.6.2 Constraints

If rootplan is not set to true, either the name or plan attribute must resolve to a valid task specified within the PRO*forma* guideline.

5.1.7 task

The *task* tag is used to represent a PRO*forma* task and provide access to its properties. Note that action, decision, enquiry and plan tasks can be used within this tag (PRO*forma* inheritance is preserved).

5.1.7.1 Attributes

- name - The name of the PRO*forma* task, as specified within the PRO*forma* guideline
- task - A scripting variable of class TaskBean
- var - An ID by which to associate the TaskBean for use within script expressions (will be of *page* scope)
- style attributes (see **Section 5.3** for further details)
- script attributes (see **Section 5.4** for further details)

5.1.7.2 Constraints

Either the name or task attribute must resolve to a valid task specified within the PRO*forma* guideline.

5.1.8 task-image

The *task-image* tag produces an HTML IMG tag corresponding to the task representation used within PRO*forma*.

5.1.8.1 Attributes

- flash *(false)* - A boolean flag used to indicate whether the image should flash if its corresponding task has a state of *in_progress*
- directory (*/images/proforma*) – The directory in which the associated .gif image files are located
- width - The width (in pixels) to use for the image
- height - The height (in pixels) to use for the image
- vspace - The vertical spacing (in pixels) to use for the image
- hspace - The horizontal spacing (in pixels) to use for the image
- style attributes (see **Section 5.3** for further details)
- script attributes (see **Section 5.4** for further details)

5.1.8.2 Constraints

This tag can only be nested within an *action, decision, enquiry, plan* or *task* tag within the Tallis or Tallis form tag libraries (see **Section 5.2**)

5.1.9 triggers

The *triggers* tag allows the author to iterate over the set of triggerable tasks within a PRO*forma* plan.

5.1.9.1 Attributes

- var (**required**) - An ID by which to associate the current triggerable TaskBean of the iteration for use within script expressions (will be of *page* scope)
- plan - A scripting variable of class PlanBean of which to return child triggerable tasks
- planName - The name of the PRO*forma* plan, as specified within the PRO*forma* guideline, of which to return child triggerable tasks
- recursive (*false*) - A flag to indicate whether triggerable tasks of any child plan should also be returned
- begin - An integer representing the starting index of the triggerable task to return in the iteration
- end - An integer representing the final index of the triggerable task to return in the iteration
- step - An integer representing the step value to use in the iteration
- status - An ID by which to associate the iteration specific javax.servlet.jsp.jstl.core.LoopTagStatus object for use within script expressions (will be of *page* scope)

5.1.9.2 Constraints

Either the planName or plan attribute must resolve to a valid plan specified within the PRO*forma* guideline.

5.1.10 trigger

The *trigger* tag produces an HTML A tag which links back to the server to allow tasks to be triggered.

5.1.10.1 Attributes

- task - A scripting variable of class TaskBean that may be triggered
- taskName - The name of the PRO*forma* task that may be triggered, as specified within the PRO*forma* guideline
- process *(/TriggerTask.do)* - The relative URL to a Struts actions that will result in the task in question being triggered
- target *(self)* - The frame into which the result of the form submission should return

- style attributes (see **Section 5.3** for further details)
- script attributes (see **Section 5.4** for further details)

5.1.10.2 Constraints

Either the taskName or task attribute must resolve to a valid, triggerable task specified within the PRO*forma* guideline.

5.1.11 components

The *components* tag allows the author to iterate over the set of task components within a PRO*forma* plan.

5.1.11.1 Attributes

- var (**required**) - An ID by which to associate the current TaskBean of the iteration for use within script expressions (will be of *page* scope)
- plan - A scripting variable of class PlanBean
- planName - The name of the PRO*forma* plan, as specified within the PRO*forma* guideline
- recursive (*false*) - A flag to indicate whether tasks of any child plan should also be returned
- tasks *(all)* - A comma seperated list of the tasks with the specified type (actions, decision, enquiry, plan, task) that will be returned during the iteration
- states *(in_progress)* - A comma seperated list of the tasks with specified state (dormant, in_progress, completed, discarded) that will be returned during the iteration
- begin - An integer representing the starting index of the triggerable task to return in the iteration
- end - An integer representing the final index of the triggerable task to return in the iteration
- step - An integer representing the step value to use in the iteration
- status - An ID by which to associate the iteration specific javax.servlet.jsp.jstl.core.LoopTagStatus object for use within script expressions (will be of *page* scope)

5.1.11.2 Constraints

Either the planName or plan attribute must resolve to a valid plan specified within the PRO*forma* guideline.

5.1.12 antecedents

The *antecedents* tag allows the author to iterate over the set of antecedent tasks of another PRO*forma* task.

5.1.12.1 Attributes

- var (**required**) - An ID by which to associate the current TaskBean of the iteration for use within script expressions (will be of *page* scope)
- task - A scripting variable of class TaskBean
- taskName - The name of the PRO*forma* task, as specified within the PRO*forma* guideline

5.1.12.2 Constraints

Either the taskName or task attribute must resolve to a valid task specified within the PRO*forma* guideline.

5.1.13 svg-plan

The *svg-plan* tag produces an HTML EMBED tag allowing PRO*forma* plans to be rendered by a SVG plugin.

5.1.13.1 Attributes

- plan - A scripting variable of class PlanBean
- planName - The name of the PRO*forma* plan, as specified within the PRO*forma* guideline
- path *(/PlanSVG.do)* - The relative URL to a Struts actions that will result in the generation of SVG representing the plan in question
- width *(900)* - The width attribute that will be associated with the generated HTML EMBED tag
- height *(600)* - The height attribute that will be associated with the generated HTML EMBED tag
- onload - The javascript onload function that will be associated with the generated HTML EMBED tag
- onunload - The javascript onunload function that will be associated with the generated HTML EMBED tag
- pluginPage *(http://www.adobe.com/svg/viewer/install/)* - The URL from which the ADOBE SVG plugin can be download if not present on the clinet's machine
- objectName - The name attribute that will be associated with the generated HTML EMBED tag

- style attributes (see **Section 5.3** for further details)

5.1.13.2 Constraints

Either the planName or plan attribute must resolve to a valid plan specified within the PRO*forma* guideline.

5.1.14 schedule-constraint

The *schedule-constraint* tag produces the SVG xml necessary to represent a schedule constraint between to PRO*forma* tasks.

5.1.14.1 Attributes

- antecedent - A scripting variable of class TaskBean acting as the arrow tail
- antecedentName - The name of the PRO*forma* task, as specified within the PRO*forma* guideline, acting as the arrow tail
- scheduled - A scripting variable of class TaskBean acting as the arrow head
- scheduledName - The name of the PRO*forma* task, as specified within the PRO*forma* guideline, acting as the arrow head
- offset *(20)* - The offset (in pixels) of the arrow head and tail from the centre points of the antecedent and scheduled tasks
- width *(2)* - The width of the arrow in pixels
- head *(3)* - The size of the arrow head in pixels
- colour *(black)* - The colour of the arrow

5.1.14.2 Constraints

Either the antecedentName or antecedent attribute must resolve to a valid task specified within the PRO*forma* guideline. Either the scheduledName or scheduled attribute must resolve to a valid task specified within the PRO*forma* guideline.

Note, no check is made to ensure that the tasks in question have an actual schedule-constraint relation within the PRO*forma* guideline.

5.1.15 dataitems

The *dataitems* tag allows the author to iterate over the set of dataitems within a PRO*forma* guideline.

5.1.15.1 Attributes

- var (**required**) - An ID by which to associate the current DataitemBean of the iteration for use within script expressions (will be of *page* scope)

- begin - An integer representing the starting index of the DataitemBean to return in the iteration
- end - An integer representing the final index of the DataitemBean to return in the iteration
- step - An integer representing the step value to use in the iteration
- status - An ID by which to associate the iteration specific javax.servlet.jsp.jstl.core.LoopTagStatus object for use within script expressions (will be of *page* scope)

5.1.15.2 Constraints

None

5.1.16 dataitem

The *dataitem* tag is used to represent a PRO*forma* dataitem and provide access to its properties.

5.1.16.1 Attributes

- var - An ID by which to associate the DataitemBean for use within script expressions (will be of *page* scope)
- name - The name of the PRO*forma* dataitem, as specified within the PRO*forma* guideline
- dataitem - A scripting variable of class DataitemBean
- output *(false)* – A boolean flag to indicate if the value of the dataitem in question should be outputted. If true, any body content within the tag is ignored
- style attributes (see **Section 5.3** for further details)
- script attributes (see **Section 5.4** for further details)

5.1.16.2 Constraints

Either the name or dataitem attribute must resolve to a valid dataitem specified within the PRO*forma* guideline.

5.1.17 datavalue

The *datavalue* tag produces an HTML read-only INPUT tag with a value set to the POR*forma* dataitem in question.

5.1.17.1 Attributes

- style attributes (see **Section 5.3** for further details)
- script attributes (see **Section 5.4** for further details)

5.1.17.2 Constraints

The *datavalue* tag MUST be nested within a valid *dataitem* tag.

5.1.18 sources

The *sources* tag tag allows the author to iterate over the set of sources within a PRO*forma* enquiry or decision.

5.1.18.1 Attributes

- var (**required**) - An ID by which to associate the current SourceBean of the iteration for use within script expressions (will be of *page* scope)
- begin - An integer representing the starting index of the SourceBean to return in the iteration
- end - An integer representing the final index of the SourceBean to return in the iteration
- step - An integer representing the step value to use in the iteration
- status - An ID by which to associate the iteration specific javax.servlet.jsp.jstl.core.LoopTagStatus object for use within script expressions (will be of *page* scope)

5.1.18.2 Constraints

The *sources* tag MUST be nested within a valid *enquiry* or *decision* tag (from either this library or the Tallis Forms library – see **Section 5.2**).

5.1.19 source

The *source* tag is used to represent a PRO*forma* source and provide access to its properties.

5.1.19.1 Attributes

- var - An ID by which to associate the SourceBean for use within script expressions (will be of *page* scope)
- name - The name of the PRO*forma* enquiry's source, as specified within the PRO*forma* guideline
- source - A scripting variable of class SourceBean
- style attributes (see **Section 5.3** for further details)
- script attributes (see **Section 5.4** for further details)

5.1.19.2 Constraints

Either the name or source attribute must resolve to a valid enquiry's source object specified within the PRO*forma* guideline.

5.1.20 candidates

The *candidates* tag tag allows the author to iterate over the set of candidates within a PRO*forma* decision.

5.1.20.1 Attributes

- var (**required**) - An ID by which to associate the current CandidateBean of the iteration for use within script expressions (will be of *page* scope)
- select - An option to specifiy the type of candidates (recommended, neutral, not-recommended) that will be returned during the iteration. If no value is specified, all candidates will be returned
- begin - An integer representing the starting index of the CandidateBean to return in the iteration
- end - An integer representing the final index of the CandidateBean to return in the iteration
- step - An integer representing the step value to use in the iteration
- status - An ID by which to associate the iteration specific javax.servlet.jsp.jstl.core.LoopTagStatus object for use within script expressions (will be of *page* scope)

5.1.20.2 Constraints

The *candidates* tag MUST be nested within a valid *decision* tag (from either this library or the Tallis Forms library – see **Section 5.2**).

5.1.21 candidate

The *candidate* tag is used to represent a PRO*forma* candidate and provide access to its properties.

5.1.21.1 Attributes

- var - An ID by which to associate the CandidateBean for use within script expressions (will be of *page* scope)
- name - The name of the PRO*forma* decision's candidate, as specified within the PRO*forma* guideline
- candidate - A scripting variable of class CandidateBean
- style attributes (see **Section 5.3** for further details)
- script attributes (see **Section 5.4** for further details)

5.1.21.2 Constraints

Either the name or candidate attribute must resolve to a valid decision's candidate object specified within the PRO*forma* guideline.

5.1.22 arguments

The *arguments* tag tag allows the author to iterate over the set of arguments within a PRO*forma* candidate.

5.1.22.1 Attributes

- var (**required**) - An ID by which to associate the current ArgumentBean of the iteration for use within script expressions (will be of *page* scope)
- select - An option to specifiy the type of arguments (confirming, for, neutral, against, excluding) that will be returned during the iteration. If no value is specified, all arguments will be returned
- begin - An integer representing the starting index of the ArgumentBean to return in the iteration
- end - An integer representing the final index of the ArgumentBean to return in the iteration
- step - An integer representing the step value to use in the iteration
- status - An ID by which to associate the iteration specific javax.servlet.jsp.jstl.core.LoopTagStatus object for use within script expressions (will be of *page* scope)

5.1.22.2 Constraints

The *arguments* tag MUST be nested within a valid *candidate* tag.

5.1.23 argument

The *argument* tag is used to represent a PRO*forma* argument and provide access to its properties.

5.1.23.1 Attributes

- var - An ID by which to associate the ArgumentBean for use within script expressions (will be of *page* scope)
- name - The name of the PRO*forma* candidate's argument, as specified within the PRO*forma* guideline
- argument - A scripting variable of class ArgumentBean
- style attributes (see **Section 5.3** for further details)
- script attributes (see **Section 5.4** for further details)

5.1.23.2 Constraints

Either the name or argument attribute must resolve to a valid candidate's argument object specified within the PRO*forma* guideline.

5.1.24 procedure

The *procedure* tag will output the procedure property of a PRO*forma* action.

5.1.24.1 Attributes

- style attributes (see **Section 5.3** for further details)
- script attributes (see **Section 5.4** for further details)

5.1.24.2 Constraints

The *procedure* tag MUST be nested within a valid *action* tag (from either this library or the Tallis Forms library – see **Section 5.2**).

5.1.25 context

The *context* tag will output the context property of a PRO*forma* task.

5.1.25.1 Attributes

- style attributes (see **Section 5.3** for further details)
- script attributes (see **Section 5.4** for further details)

5.1.25.2 Constraints

The *context* tag MUST be nested within a valid *action, decision, enquiry, plan* or *task* tag (from either this library or the Tallis Forms library – see **Section 5.2**).

5.1.26 caption

The *caption* tag will output the caption property of a PRO*forma* component.

5.1.26.1 Attributes

- style attributes (see **Section 5.3** for further details)
- script attributes (see **Section 5.4** for further details)

5.1.26.2 Constraints

The *caption* tag MUST be nested within a valid *action, decision, enquiry, plan, task* (from either this library or the Tallis Forms library – see **Section 5.2**), *candidate, argument, source* or *dataitem* tag.

5.1.27 description

The *description* tag will output the caption property of a PRO*forma* component.

5.1.27.1 Attributes

- style attributes (see **Section 5.3** for further details)

- script attributes (see **Section 5.4** for further details)

5.1.27.2 Constraints

The *description* tag MUST be nested within a valid *action, decision, enquiry, plan, task* (from either this library or the Tallis Forms library – see **Section 5.2**), *candidate, argument, source* or *dataitem* tag.

5.2 Tallis forms

The Tallis Forms tag library provides **read/write** functionality to a subset of PRO*forma* components within the engine. The library is defined in the file *TOMCAT_HOME*/webapps/*WEB_APP*/WEB-INF/tld/tallis-forms.tld (see Appendix A for details) and can be imported into your JSP with the URI http://acl.icnet.uk/tallis-forms.

5.2.1 action

The *action* tag produces an HTML FORM tag with a hidden field (taskName), whose value is the name of the PRO*forma* action being processed.

5.2.1.1 Attributes

- var - An ID by which to associate the ActionBean for use within script expressions (will be of *page* scope)
- name - The name of the PRO*forma* action, as specified within the PRO*forma* guideline
- action - A scripting variable of class ActionBean
- process *(/ConfirmTask.do)* - The relative URL to a Struts actions that will result in the submission of the action to the server for confirmation
- formName - The name attribute that will be associated with the generated HTML FORM tag
- target *(self)* - The frame into which the result of the form submission should return
- onreset - The javascript onreset function that will be associated with the generated HTML FORM tag
- onsubmit - The javascript onsubmit function that will be associated with the generated HTML FORM tag
- style attributes (see **Section 5.3** for further details)

5.2.1.2 Constraints

Either the name or action attribute must resolve to a valid action specified within the PRO*forma* guideline.

5.2.2 decision

The *decision* tag produces an HTML FORM tag with a hidden field (taskName), whose value is the name of the PRO*forma* decision being processed.

5.2.2.1 Attributes

- var - An ID by which to associate the DecisionBean for use within script expressions (will be of *page* scope)
- name - The name of the PRO*forma* decision, as specified within the PRO*forma* guideline
- decision - A scripting variable of class DecisionBean
- process *(/ConfirmTask.do)* - The relative URL to a Struts actions that will result in the submission of the decision to the server for confirmation
- formName - The name attribute that will be associated with the generated HTML FORM tag
- target *(self)* - The frame into which the result of the form submission should return
- onreset - The javascript onreset function that will be associated with the generated HTML FORM tag
- onsubmit - The javascript onsubmit function that will be associated with the generated HTML FORM tag
- style attributes (see **Section 5.3** for further details)

5.2.2.2 Constraints

Either the name or decision attribute must resolve to a valid decision specified within the PRO*forma* guideline.

5.2.3 enquiry

The *enquiry* tag produces an HTML FORM tag with a hidden field (taskName), whose value is the name of the PRO*forma* enquiry being processed.

5.2.3.1 Attributes

- var - An ID by which to associate the EnquiryBean for use within script expressions (will be of *page* scope)
- name - The name of the PRO*forma* enquiry, as specified within the PRO*forma* guideline
- enquiry - A scripting variable of class EnquiryBean
- process *(/ConfirmTask.do)* - The relative URL to a Struts actions that will result in the submission of the enquiry to the server for confirmation

- formName - The name attribute that will be associated with the generated HTML FORM tag
- target *(self)* - The frame into which the result of the form submission should return
- onreset - The javascript onreset function that will be associated with the generated HTML FORM tag
- onsubmit - The javascript onsubmit function that will be associated with the generated HTML FORM tag
- style attributes (see **Section 5.3** for further details)

5.2.3.2 Constraints

Either the name or enquiry attribute must resolve to a valid enquiry specified within the PRO*forma* guideline.

5.2.4 task

The *task* tag produces an HTML FORM tag with a hidden field (taskName), whose value is the name of the PRO*forma* task being processed. Note that action, decision and enquiry tasks can be used within this tag (PRO*forma* inheritance is preserved).

5.2.4.1 Attributes

- var - An ID by which to associate the TaskBean for use within script expressions (will be of *page* scope)
- name - The name of the PRO*forma* task, as specified within the PRO*forma* guideline
- task - A scripting variable of class TaskBean
- process *(/ConfirmTask.do)* - The relative URL to a Struts actions that will result in the submission of the task to the server for confirmation
- formName - The name attribute that will be associated with the generated HTML FORM tag
- target *(self)* - The frame into which the result of the form submission should return
- onreset - The javascript onreset function that will be associated with the generated HTML FORM tag
- onsubmit - The javascript onsubmit function that will be associated with the generated HTML FORM tag
- style attributes (see **Section 5.3** for further details)

5.2.4.2 Constraints

Either the name or task attribute must resolve to a valid task specified within the PRO*forma* guideline.

5.2.5 source-field

The *source-field* tag produces an HTML INPUT tag with a value set to the POR*forma* dataitem referenced by the source in question.

5.2.5.1 Attributes

- dropdown *(false)* - A boolean flag to indicate whether a range of possible value should be displayed as a drop menu or radio buttons
- hidden *(false)* - A boolean flag to indicate whether the generated HTML INPUT tag should have a type of hidden
- style attributes (see **Section 5.3** for further details)
- script attributes (see **Section 5.4** for further details)

5.2.5.2 Constraints

The *source-field* tag MUST be nested within a valid *source* tag.

5.2.6 datavalue

The *datavalue* tag produces an HTML INPUT tag with a value set to the POR*forma* dataitem in question.

5.2.6.1 Attributes

- dropdown *(false)* - A boolean flag to indicate whether a range of possible value should be displayed as a drop menu or radio buttons
- readonly *(false)* - A boolean flag to indicate whether the generated HTML INPUT tag should be read-only
- disabled *(false)* - A boolean flag to indicate whether the generated HTML INPUT tag should be set as disbaled
- style attributes (see **Section 5.3** for further details)
- script attributes (see **Section 5.4** for further details)

5.2.6.2 Constraints

The *datavalue* tag MUST be nested within a valid *dataitem* tag.

5.2.7 candidate-field

The *candidate-field* tag produces an HTML INPUT tag, allowing candidates to be committed during form submission.

5.2.7.1 Attributes

- style attributes (see **Section 5.3** for further details)
- script attributes (see **Section 5.4** for further details)

5.2.7.2 Constraints

The *candidate-field* tag MUST be nested within a valid *candidate* tag.

5.3 General style attributes

Some of the tags within the Tallis tag libraries have the facility to make use of CSS to control their rendering within the browser. Tags that support styling may make use of the following attributes:

- style – a list of CSS property/value pairs (*cf* HTML style attribute)
- styleClass – a reference to a CSS class defined within a stylesheet (*cf* HTML class attribute)
- styleId - a reference to a CSS id defined within a stylesheet (*cf* HTML id attribute)

5.4 General functional attributes

Some of the tags within the Tallis tag libraries have the facility to make use of JavaScript functions to control their behaviour within the browser (*cf* HTML intrinsic event attributes). Tags that support scripting may make use of the following attributes:

- onblur
- onchange
- onclick
- ondblclick
- onfocus
- onkeydown
- onkeypress
- onkeyup
- onmousedown
- onmousemove
- onmouseout
- onmouseover

- onmouseup

6. Appendices

6.1 Appendix A

When the Tallis web enactment components are running locally on your machine, to access the application you would use the URL http://localhost:8080/*WEB_APP*. This *BASE_URL* is used to resolve all application resources.

Resources marked with a leading slash (ex. /images/hello.gif) are resolved using the *BASE_URL* (i.e. the .gif file MUST exist in the Tomcat application with the path *WEB_APP*/images/hello.gif).

If the *tallis.directory* property starts with a leading slash (ex. /customisations) and a *tallis.*page is specified (ex. /pages/include.jsp) the above resolution rule also applies (i.e. the resource MUST exist in the Tomcat application with the path *WEB_APP*/customisations/pages/include.jsp).

These paths resolve to a file system path of *TOMCAT_HOME*/*WEB_APP*/*path*, where *TOMCAT_HOME* is the path to your Tomcat installation.

6.2 Appendix B

Useful URLs to Java serverside technologies include:

- Java Servlets - http://java.sun.com/products/servlet/index.html
- Java Server Pages (JSP) - http://java.sun.com/products/jsp/
- Tab libraries - http://java.sun.com/products/jsp/taglibraries.html
- JSTL - http://java.sun.com/products/jsp/jstl/index.html
- Tomcat - http://jakarta.apache.org/tomcat/index.html
- Struts - http://jakarta.apache.org/struts/index.html
- Tiles - http://www.lifl.fr/~dumoulin/tiles/index.html