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-2005-007

Customising the Standard Web Enactment Interface

Ayelet Oettinger & Rory Steele

Creation date	04 October 2005
Last modification	21 October 2005
Last edited by	Ayelet Oettinger
Version	1
Circulation	unrestricted
Status	unreviewed

1. Introduction 2

2. The Mapping File 2

2.1 File Structure 2

2.2 Location 7

2.3 Pointing to the Mapping File 7

3. Customised Task Pages 7

3.1 Creating a Customised Task Page 7

3.2 Required Content for Customised Task Pages 8

4. The Folder Hierarchy 9

5. Enacting a Customised Process-Description Over the Web 10

6. Creating a Tallis Web Enactment Application 10

6.1 Content Changes 10

6.2 Style Changes 11

6.3 Creating a war file 12

7. Appendix 1: Mapping File XML Schema 12

1. Introduction

The Tallis Web Enactment application dynamically generates standard web pages that take the end-user through a process-description's component tasks.

In most cases this is adequate, but if required, the standard web pages can be replaced by web pages customised for an individual process-description. 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 process-description, such as images, links to further information and supporting material, widgets and even applets to aid the user during the enactment process. Using standard JSP tag libraries, it is also possible to perform additional functions at the server, such as automated database queries and mail processing.

Customising the standard web enactment interface can involve as little as modifying some task attributes in a XML mapping file. Otherwise, customised pages can be created for specific tasks, supplementing the standard html page with customised content or with an applet. Apart from modifying specific task pages, it is also possible to modify the web application itself. This document provides an introduction to the customisation of standard task pages. For a more detailed account of web-enactment customisation see Tallis Interface Primer.

2. The Mapping File

Certain PROforma 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 PROforma content.

The XML mapping file conforms to the 'http://www.cancerresearchuk.org/acl/proforma/content#' schema. If you have downloaded the Tallis Web Enactment suite, you can find the schema in \Apache Software Foundation\Tomcat 5.0\webapps\WEB_APP\WEB-INF\PRO*forma* Content.xsd (or see appendix 1).

2.1 File Structure

2.1.1 Guideline Element

The content of the mapping file is contained within the *guideline* tag:

```
<?xml version="1.0" encoding="UTF-8"?>
<pf:guideline
xmlns:pf="http://www.cancerresearchuk.org/acl/proforma/conte
nt#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.cancerresearchuk.org/acl/prof
orma/content# /WEB-INF/proformaContent.xsd">
 CONTENT_OF_MAPPING_FILE
</pf:guideline>
```

2.1.2 Task Elements

Five elements can populate the *guideline* tag: action, decision, enquiry, plan and task (i.e., keystone).

Customisations for each task are defined within task definition tags, which identify the specific task by adding the value of its PRO*forma* Task Name to the *definition* attribute, e.g.:

```
<pf:action definition="action_209019">
   CONTENT_OF_ACTION_CUSTOMISATION
</pf:action>
```

A *URI* attribute, which specifies a specific page of content to be used rather than the automatically generated content, can be added to the task definition tag:

```
<pf:action definition="action_209019"
uri="/customisations/pages/mappingTest/action.jsp">
   CONTENT_OF_ACTION_CUSTOMISATION
</pf:action>
```

Customisations in a separate file override customisations in the mapping file. For more information about customised task pages see section 3.

Another useful attribute that can be added to each task is *stylesheet*. It specifies a specific style sheet to be used rather than the standard runtime style sheet. Tasks inherit the *stylesheet* attribute of their parent-plan.

```
<pf:action definition="action_209019"
stylesheet="/customisations/stylesheets/special_action.css">
</pf:action>
```

2.1.3 Customising PRO*forma* Entities

The caption and the description of PRO*forma* entities can be modified by using the mapping file.

2.1.3.1 Caption & Description

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

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

Both tags MUST be nested within a valid *action*, *decision*, *enquiry*, *plan*, *task*, *candidate*, *argument*, or *source* tag. They can contain any valid XML content.

Example:

```
<pf:action definition="action_209019">
   <pf:caption>TASK_CAPTION</pf:caption>
   <pf:description>TASK_DESCRIPTION</pf:description>
</pf:action>
```

2.1.4 Customising Tasks

A task's caption and description can be modified as described above.

Additionally, the task's context can be modified by using the *context* tag. The *context* tag can contain any valid XML content.

2.1.5 Customising Actions

In addition to the base task properties described above, an action's procedure can be modified by using the *procedure* tag. The *procedure* tag can contain any valid XML content.

The *delegate* attribute of the *action* tag can be used to group and display a set of actions on one page, by pointing to a specified PRO*forma* plan. This is described in section 2.1.8 below.

2.1.6 Customising Enquiries

The mapping file enables users to control the order of the sources displayed for an enquiry and the user interface widget used for each source, and to introduce dependencies between sources.

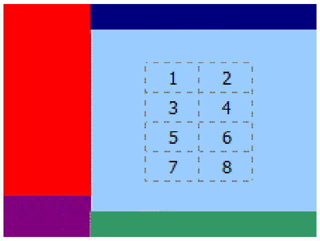
The enquiry task definition tag can contain a *sources* tag – which contains at least one *source* tag – and one or more *dependency* tags.

The *delegate* attribute of the *enquiry* tag can be used to group and display a set of enquiries on one page, by pointing to a specified PRO*forma* plan. This is described in section 2.1.8 below.

2.1.6.1 Source Tag

The *source* tag has three attributes:

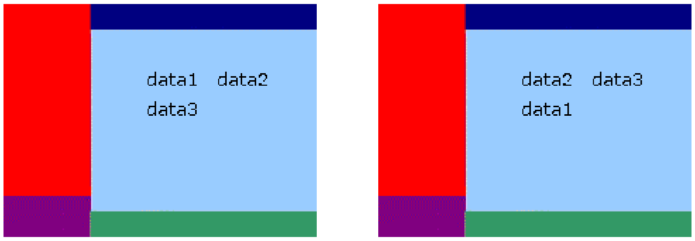
- *definition* – identifies a PRO*forma* source by its Name value
- *position* – an integer which defines the position of the source on the page, according to the following order:



- *dropdown* – determines whether the source values are displayed using radio-buttons (or checkboxes in the case of multiple selection of values) or a dropdown list. By default, checkboxes or radio-buttons are used (dropdown=false).

Example:

```
<pf:sources>
 <pf:source definition="data1" position="3"
dropdown="true"/>
 <pf:source definition="data2" position="1"/>
 <pf:source definition="data3" position="2"/>
</pf:sources>
```



On the left is the default screen layout; on the right is the screen layout using the above mapping file customisation.

2.1.6.2 Dependency Tag

This tag determines dependencies between sources. The tag defines a *source* attribute and a *keyword* value for that source, for which the nested source would be disabled, e.g.:

```
<pf:dependency source="sex"
keyword="male">pregnancy</pf:dependency>
```

The `Pregnancy` source is disabled if the `male` value is selected for the `sex` source.

2.1.7 Customising Decisions

If a decision has sources, it can be customised just like an enquiry (see section 2.1.6 above). Additionally, captions and descriptions of decisions, candidates and arguments can be modified using the relevant tags (see section 2.1.3 above).

Each *decision* tag can have one *candidates* tag. This tag must contain at least one *candidate* tag. A *candidate* tag can contain one *arguments* tag. This tag must contain at least one *argument* tag.

The *maxRecommendations* attribute of the *decision* tag can be used to determine the maximum number of candidates to be presented to the end-user as recommended, irrespective of how many are actually recommended by the enactment engine.

Example:

```
<pf:decision definition="decision_109238"
maxRecommendations="1">
 <pf:caption>A Decision</pf:caption>
 <pf:candidates>
   <pf:candidate definition="cand1">
```

```
     <pf:caption>Candidate C1</pf:caption>
     <pf:description>Some description for candidate
1</pf:description>
     <pf:arguments>
       <pf:argument definition="arg1">
         <pf:caption>Argument 1</pf:caption>
         <pf:description>Some description for argument
1</pf:description>
       </pf:argument>
       <pf:argument definition="arg2">
         <pf:caption>Argument 2</pf:caption>
         <pf:description>Some description for argument
2</pf:description>
       </pf:argument>
       <pf:argument definition="arg3">
         <pf:caption>Argument 3</pf:caption>
         <pf:description>Some description for argument
3</pf:description>
       </pf:argument>
     </pf:arguments>
   </pf:candidate>
   <pf:candidate definition="cand2">
     <pf:caption>Candidate C2</pf:caption>
     <pf:arguments>
       <pf:argument definition="arg4">
         <pf:caption>Argument 4</pf:caption>
       </pf:argument>
       <pf:argument definition="arg5">
         <pf:caption>Argument 5</pf:caption>
       </pf:argument>
     </pf:arguments>
   </pf:candidate>
 </pf:candidates>
</pf:decision>
```

Note: In the current version of Tallis Composer (1.4) arguments do not have a Name field. Instead, the argument's condition is automatically entered as argument name in the PRO*forma* definition. This makes the use of the *argument* tag cumbersome and error-prone.

2.1.8 Customising Plans

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

The plan has to point to a customised plan page that will display the multiple enquiries (if you have downloaded the Tallis Web Enactment suite, you can find a customised plan page for enquiries (enquiryGroup.jsp) and one for actions (actionGroup.jsp) in \Apache Software Foundation\Tomcat 5.0\webapps\WEB_APP\customisations\pages\).

Example:

```
<pf:enquiry definition="enquiry1" delegate="plan1"/>
<pf:enquiry definition="enquiry2" delegate="plan1"/>
<pf:enquiry definition="enquiry3" delegate="plan1"/>
<pf:plan definition="plan1" uri="/customisations/pages/enquiryGroup.jsp">
 <pf:grouping>enquiry1 enquiry2 enquiry3</pf:grouping>
</pf:plan>
```

2.2 Location

The mapping file has to be placed within the web enactment application folder, e.g.:

```
webapps/tallis/MappingFile.xml
```

OR

```
webapps/tallis/customisations/mappings/MappingFile.xml
```

2.3 Pointing to the Mapping File

The customised process-description has to point to the mapping file. This is done via the Context field of the top-level plan.

In the Context field of the top-level plan type:

```
#metadata
tallis.content = PATH_TO_MAPPING_FILE
```

The path is relative to the web application, e.g:

```
#metadata
tallis.content = /customisations/mappings/mappingFile.xml
```

3. Customised Task Pages

Although most customisations can be done using the mapping file, there are several cases where it is advisable to use separate customised task pages:

- When adding dynamic content, as this can only be achieved via a JSP file.
- When there are many customised tasks, each with a lot of content, as the mapping file can become large and less efficient both in terms of editing and in terms of running the process.

3.1 Creating a Customised Task Page

The Tallis Web Enactment application dynamically creates standard web pages for tasks that don’t have the mappings to specify alternative content. The HTML code of these pages can be saved, edited and reused by Tallis during future enactments of the process-description.

Follow the steps below to use the dynamically generated code produced by the Tallis Web Enactment application:

- Start the enactment process: Run through the guideline until you reach the task you wish to associate with customised content.

- Capture the generated code: Right-click on the page in question and select 'view-source'.

- The dynamically generated content for each task is created within the following comment tags to allow ease of location:

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

  generated content

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

- Copy the html between the comment tags and save the resulting text.

- Edit the generated code and save it: The new file MUST be accessible to the Tallis web-application you wish to run. This does not mean that the file must be contained within the Tomcat installation, only that it can be delivered via HTTP.

- Edit the mapping file: The customised page has to be specified in the mapping file, using the *URI* attribute of the *task* tag (see section 2.1.2).

- Edit the PROforma process-description: Add the necessary context properties to the top-level plan (the mapping file has to be located and pointed to as described in sections 2.2 and 2.3).

When creating a separate customised task page, it's best to use jsp files as they are less likely to cause errors.

Customisations in a separate task file override customisations in the mapping file.

3.2 Required Content for Customised Task Pages

Specific information is required by the server for a task to be processed, with different task types requiring different information for a request to be successful. If minimal editing of the automatically generated content is performed (e.g., structural changes to layout), this is generally not an issue. However, if the content is significantly altered or has been built from scratch, this additional information may need to be explicitly managed.

3.2.1 Actions, plans and keystone tasks

The web enactment 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.

3.2.2 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 Enquiries below for further details).

3.2.3 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).

4. The Folder Hierarchy

If you have installed the Tallis Web Enactment suite, you will find the following folders in the Apache Software Foundation directory:

- **webapps**
  - **tallis** (or another web enactment application folder)
    - **customisations**

      Files for customising the web enactment of specific process-descriptions
      - **mappings**

        It is recommended to place your mapping files in this directory
      - **pages**

        It is recommended to place your customised task pages in this directory
      - **scripts**

        It is recommended to place your customised scripts in this directory
      - **stylesheets**

        It is recommended to place your customised style sheets in this directory
    - **runtime**

      JSP files for general layout and for specific components of the web application;
      These files can be used to modify the content and the look & feel of the entire web enactment application
      - **captions**

        JSP files for captions

- **confirmations**
  JSP files for confirmations
- **contents**
  JSP files for contents
- **footer**
  JSP files for the footer
- **headers**
  JSP files for headers
- **images**
  Images for the web enactment application
- **menus**
  JSP files for menus
- **scripts**
  Scripts for the web enactment application
- **stylesheets**
  Style sheets for the web enactment application

5. Enacting a Customised Process-Description Over the Web

Standard process-descriptions can be enacted over the web using the ACL server repository. In order to run a customised process-description over the web, the mapping file must be placed within the web application folder, and the customised pages (if there are any) must be accessible via HTTP. As this type of content can contain malicious code, placing such files on the ACL server cannot be done directly. Hence, if you want to enact a customised process-description over the web, either use your own server, or contact us.

6. Creating a Tallis Web Enactment Application

Start off with a working web enactment application, e.g., CustomTallis1.4.1

Copy and rename the web enactment application folder – note that this name will be part of the url for running the application, therefore avoid spaces.

6.1 Content Changes

6.1.1 Caption in Web Browser Title Bar

```
WEB-INF\conf\tiles\tiles-runtime.xml
```

Change "Tallis" in `<put name="title" value="Updating Tallis..."/>` to the name of your choice.

Note that the title should be changed for the different pages, e.g., error page, update page, etc.

6.1.2 Header caption

```
runtime\headers\header.jsp
runtime\headers\completionHeader.jsp
runtime\headers\errorHeader.jsp

<td class="headerlogocell">
  Tallis Web Enactment
</td>
```

6.1.3 Footer

```
runtime\footers\footer.jsp

<td class="footer" width="100%">
  &copy;<a href="http://acl.icnet.uk/" target="blank">
ACL</a> | <a href="http://www.cancerresearchuk.org/"
target="blank">CR-UK</a>
</td>
```

6.2 Style Changes

```
runtime\stylesheets\proforma.css
```

6.2.1 Logo & Web Enactment Title

Place the logo image in `runtime\images`

```
.headerlogo
{
  width: 100%;
  height: 40px;
  background-color: #999999;
  background-image: url(../images/Logo.gif);
  background-repeat: no-repeat;
  background-position: 2px 2px;
  font-family: Tahoma, Verdana, Arial, Helvetica, sans-
serif;
  font-size: 18px;
  font-weight: bold;
  text-indent: 36px;
  color: #f3f3f3;
}
```

6.2.2 Process title

```
.protocolName
{
  padding: 5px 10px 5px 10px;
  margin-top: 3px;
  margin-bottom: 0px;
  font-family: verdana, helvetica, arial, sans-serif;
  font-size: 13px;
  font-weight: bold;
  color: #777777;
  border: 1px #eeeeee solid;
  background-color: #eff3fb;
}
```

6.2.3 Images

Current task icon: `runtime\images\current.gif`
Save icon: `runtime\images\save.gif`
Restart icon: `runtime\images\restart.gif`
Quit icon: `runtime\images\quit.gif`
Print icon: `runtime\images\print.gif`
Logo: `runtime\images\Logo.gif`

6.3 Creating a war file

Use `war.bat` to create a `war` file of a web application folder.

6.3.1 The Code

```
cd WebAppName
C:\j2sdk1.4.2_06\bin\jar -cvf WebAppName.war .
move WebAppName.war ..
```

6.3.2 How To Use It

1. Place `war.bat` in the `webapps` directory
2. Open file with text editor
3. Replace `WebAppName` with the name of the web enactment application, e.g., `tallis1.3.5`
4. Save
5. Run `war.bat`
6. A `WebAppName.war` is created in the `webapps` directory

7. Appendix 1: Mapping File XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 4 U (http://www.xmlspy.com) by Rory
Steele (Cancer Research UK) -->
<xs:schema
xmlns="http://www.cancerresearchuk.org/acl/proforma/content#"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
targetNamespace="http://www.cancerresearchuk.org/acl/proforma/content
#" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="guideline">
    <xs:annotation>
      <xs:documentation>Comment describing your root
element</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence maxOccurs="unbounded">
        <xs:choice>
          <xs:element name="action">
            <xs:complexType>
              <xs:complexContent>
                <xs:extension base="taskContentType">
                  <xs:sequence>
                    <xs:element name="procedure"
type="contentType" minOccurs="0"/>
                  </xs:sequence>
                </xs:extension>
              </xs:complexContent>
            </xs:complexType>
          </xs:element>
          <xs:element name="decision">
            <xs:complexType>
              <xs:complexContent>
                <xs:extension base="enquiryContentType">
                  <xs:sequence>
                    <xs:element name="candidates" minOccurs="0">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="candidate"
maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:complexContent>
                                <xs:extension
base="namedContentType">
                                  <xs:sequence>
                                    <xs:element
name="arguments" minOccurs="0">
                                      <xs:complexType>
                                        <xs:sequence>
                                          <xs:element
name="argument" type="namedContentType" maxOccurs="unbounded"/>
                                        </xs:sequence>
                                      </xs:complexType>
                                    </xs:element>
                                  </xs:sequence>
                                </xs:extension>
                              </xs:complexContent>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute name="maxRecommendations"
type="xs:int" use="optional"/>
```

```
                  <xs:attribute name="comparator"
type="xs:string" use="optional"
default="org.cruk.struts.modules.shared.beans.DefaultCandidateCompara
tor"/>
                </xs:extension>
              </xs:complexContent>
            </xs:complexType>
          </xs:element>
          <xs:element name="enquiry" type="enquiryContentType"/>
          <xs:element name="plan">
            <xs:complexType>
              <xs:complexContent>
                <xs:extension base="taskContentType">
                  <xs:sequence>
                    <xs:element name="grouping"
type="componentNamesType" minOccurs="0"/>
                  </xs:sequence>
                </xs:extension>
              </xs:complexContent>
            </xs:complexType>
          </xs:element>
          <xs:element name="task">
            <xs:complexType>
              <xs:complexContent>
                <xs:extension base="taskContentType"/>
              </xs:complexContent>
            </xs:complexType>
          </xs:element>
        </xs:choice>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="enquiryContentType">
    <xs:complexContent>
      <xs:extension base="taskContentType">
        <xs:sequence>
          <xs:element name="sources" minOccurs="0">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="source" maxOccurs="unbounded">
                  <xs:complexType>
                    <xs:complexContent>
                      <xs:extension base="namedContentType">
                        <xs:attribute name="position"
type="xs:int" use="optional"/>
                        <xs:attribute name="dropdown"
type="xs:boolean" use="optional" default="false"/>
                      </xs:extension>
                    </xs:complexContent>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="dependency" minOccurs="0"
maxOccurs="unbounded">
            <xs:complexType>
```

```
              <xs:simpleContent>
                <xs:extension base="componentNamesType">
                  <xs:attribute name="source" type="xs:string"
use="required"/>
                  <xs:attribute name="keyword" type="xs:string"
use="required"/>
                </xs:extension>
              </xs:simpleContent>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="contentType" mixed="true">
    <xs:sequence>
      <xs:any namespace="##any" processContents="skip" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="namedContentType">
    <xs:sequence>
      <xs:element name="caption" type="contentType" minOccurs="0"/>
      <xs:element name="description" type="contentType"
minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="definition" type="xs:string"
use="required"/>
  </xs:complexType>
  <xs:complexType name="taskContentType">
    <xs:complexContent>
      <xs:extension base="namedContentType">
        <xs:sequence>
          <xs:element name="context" type="contentType"
minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="uri" type="xs:anyURI" use="optional"/>
        <xs:attribute name="script" type="xs:string"
use="optional"/>
        <xs:attribute name="stylesheet" type="xs:string"
use="optional"/>
        <xs:attribute name="confirmation" type="xs:string"
use="optional" default="confirmTask"/>
        <xs:attribute name="delegate" type="xs:string"
use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:simpleType name="componentNamesType">
    <xs:list itemType="xs:string"/>
  </xs:simpleType>
</xs:schema>
```