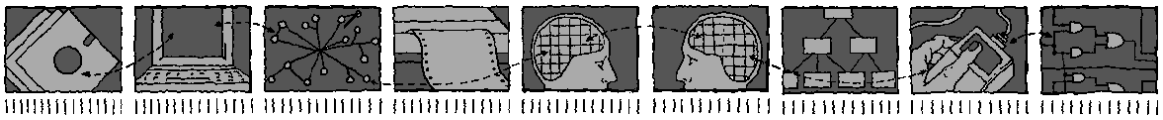
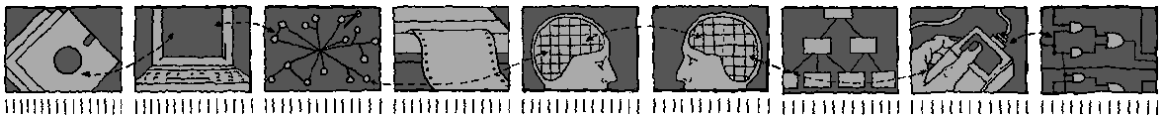


Department of Computing Science and Mathematics
University of Stirling



The ACCENT Policy Wizard

Kenneth J. Turner

Technical Report CSM-166

ISSN 1460-9673

September 2004 (updated December 2005)

Department of Computing Science and Mathematics
University of Stirling

The ACCENT Policy Wizard

Kenneth J. Turner

Department of Computing Science and Mathematics
University of Stirling
Stirling FK9 4LA, Scotland

Telephone +44-786-467421, Facsimile +44-786-464551
Email kjt@cs.stir.ac.uk

Technical Report CSM-166

ISSN 1460-9673

September 2004 (updated December 2005)

Abstract

The ACCENT project was concerned with developing a practical and comprehensive policy language for call control. The project therefore studied a number of distinct tasks: the definition of the language, and also a three-layer architecture for deploying and enforcing policies defined in the language. This report focuses on a policy wizard that acts as the primary interface between end users and the policy system. The policy wizard has an intimate knowledge of APPEL (the ACCENT Project Policy Environment/Language). This allows end users to create policies without knowing or seeing XML, and to upload them to the policy system. The wizard also provides a number of convenience functions such as pre-defined policy templates, editing and activating existing policies, and defining policy variables.

Contents

List of Figures

Chapter 1

Overview

The ACCENT project is concerned with developing a practical and comprehensive policy language for the call control domain. As such the project is concerned with a number of distinct tasks: the definition of the language, as well as the definition of a three-layer architecture for deploying and enforcing policies defined in the language. This report focuses on a policy wizard that acts as the primary interface between end users and the policy system. The policy wizard has an intimate knowledge of APPEL (the ACCENT Project Policy Environment/Language). This allows end users to create policies without knowing or seeing XML, and to upload them to the policy system. The wizard also provides a number of convenience functions such as pre-defined policy templates, editing and activating existing policies, and defining policy variables.

The reader should consider other supporting documentation, in particular [1, 2]. Technical reports describe the ACCENT Project Policy Environment/Language ([4]) and the ACCENT Policy Server ([3]).

Chapter 2

Policy Wizard User Interface

2.1 General Principles

The policy wizard is not a wizard in the sense of a program that takes the user through a well-defined task such as creating a form letter or an Internet connection. However it is a wizard in that it provides a user-friendly interface to a complex technical task (defining policies in XML form). The wizard is the primary interface to the policy system for ordinary end users. Important aspects of its design include:

- The wizard is web-based. This means that it can be used from anywhere, including away from the user's normal base. The possibility of a voice-based wizard has been considered (using VoiceXML) but not yet implemented.
- The wizard is multi-lingual (currently English, French and German). This allows the user to use the policy system irrespective of the user's preferred language.
- The wizard supports multiple levels of expertise (beginner, intermediate, expert, administrator). This allows a beginning user to see the minimum of the policy language, but an expert to see the full depth of its capabilities.
- The wizard has extensive help when defining policies. When a field has to be filled in, hints are provided. Hovering over a link provides a 'tool tip'. Online help is also provided in the user's preferred language.

Note that the policy wizard does not currently support resolution policies. They must therefore be prepared manually in XML and added directly through the policy server GUI.

2.2 Screen Shots

Since the wizard is designed to be user-friendly, little needs to be done here to explain the interface. The wizard does not support 'undo'. It is therefore suggested that policies be created and checked step-by-step. If a major error is made during editing, click Cancel and start again.

- The login screen in figure 2.1 is straightforward.
- The main menu in figure 2.2 shows 'Edit Users' only if an administrator has logged in.
- Choosing 'Existing Policy' leads to figure 2.3, where an existing policy can be selected for editing, enabling, disabling, or deletion.
- Choosing 'From Template' leads to figure 2.4, where a number of pre-defined policies can be selected and edited. Note that a blank policy is created from a special template. Templates may contain values prefixed by '?'; these must be filled in by the user before the policy is saved.

Figure 2.1: Login Screen

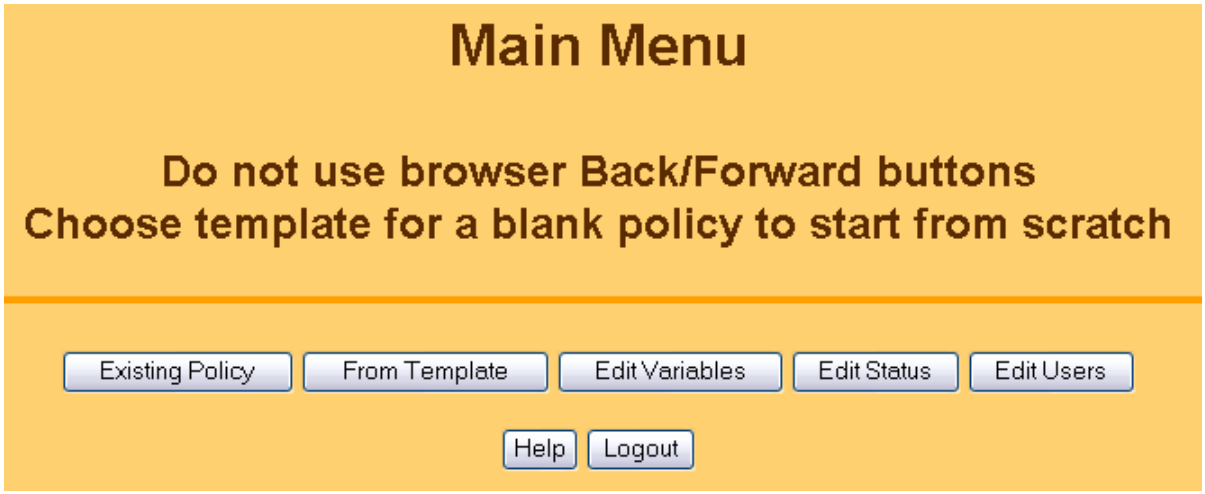


Figure 2.2: Main Menu

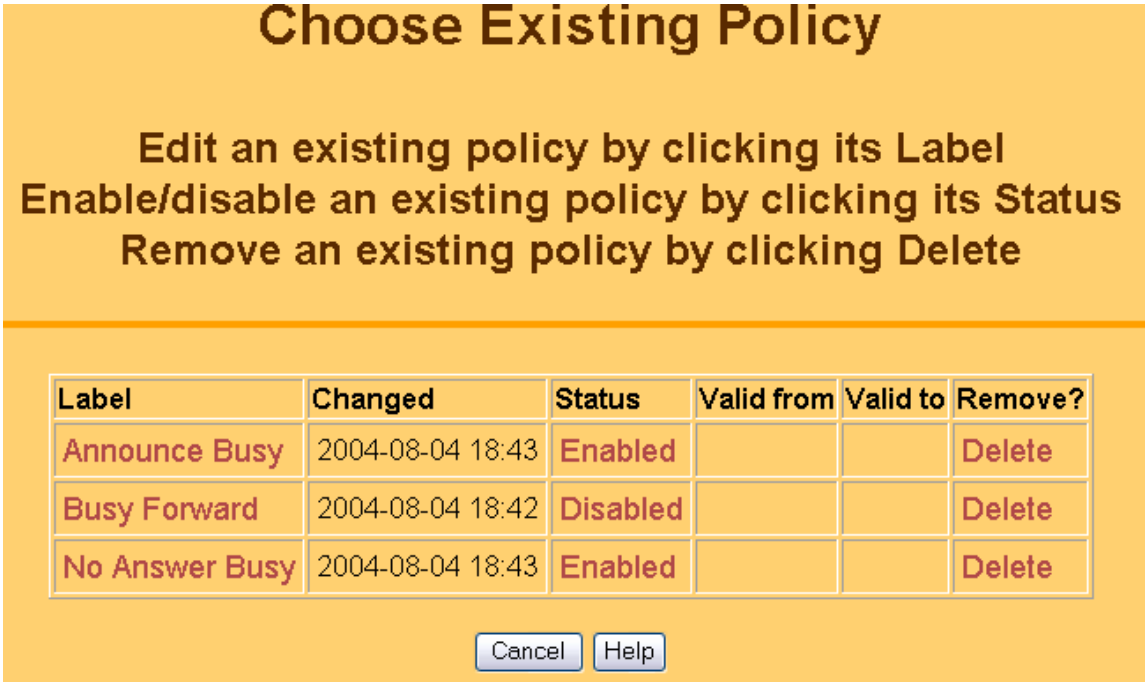


Figure 2.3: Existing Policies

- Choosing 'Edit Variables' leads to figure 2.5, where an existing variable can be selected for editing or deletion. Variables are created in text or audio clip format. Editing an audio clip is illustrated in figure 2.6. This provides record (red circle), stop (blue square) and play (green triangle) buttons. The meter bar shows the audio level during recording or the progress through the clip while playing.

- Choosing 'Edit Status' leads to figure 2.7, where the user's availability, presence and profile can be edited. There are simple check boxes (ticks) for setting availability and presence. Alternatively, specific values can be filled in for these to indicate topics that the user is willing to be called about and the user's location. The profile is used to enable a group of policies quickly.

- Choosing 'Edit Users' leads to figure 2.8, where user accounts can be created, edited or deleted. When a new user is created, policy variables are set to indicate the user is unavailable, absent and has an empty profile. When a user is deleted, all policies and variables owned by this user are removed. The 'admin' account is required and cannot be deleted, although it can be edited.

- Choosing an existing or template policy leads to figure 2.9, where the policy can be edited. The applicability, preference and rules of a policy are independently edited. The only complex aspect is adding or deleting parts of a rule. The '$\cdots$' symbol after a trigger, condition or action allows a further trigger, condition or action to be added with a specified combination. The '$\cdots$' symbol after a rule allows a further rule to be added with a specified combination. In this way, complex tree structures can be created. Figure 2.9 shows the 'append condition' tool tip resulting from hovering over the '$\cdots$' symbol after the first condition.

  To remove the first part of a combination (trigger, condition or action), set it blank. To remove the second part of a combination, click on the name of the combination and set it blank.

Figure 2.4: Template Policies

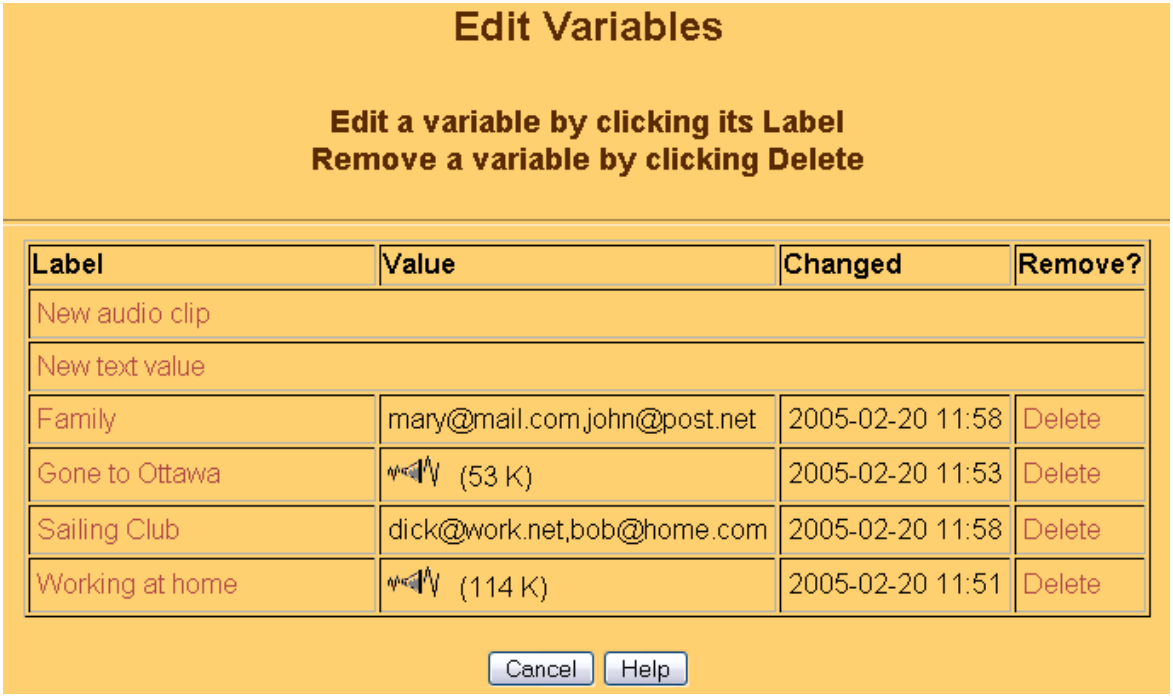


Figure 2.5: Edit Variables

Figure 2.6: Edit Audio Clip

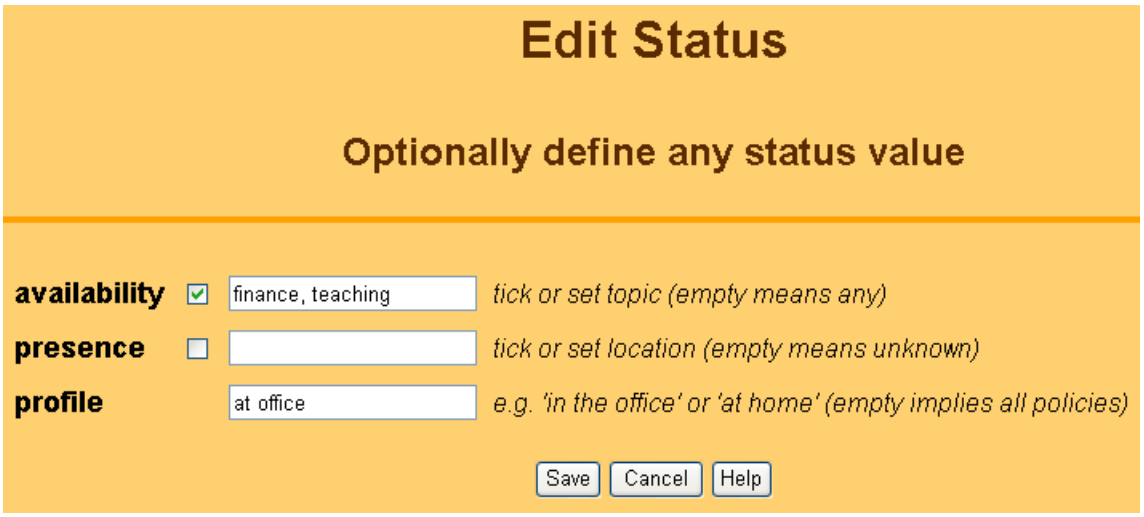


Figure 2.7: Edit Status

Figure 2.8: Edit Users

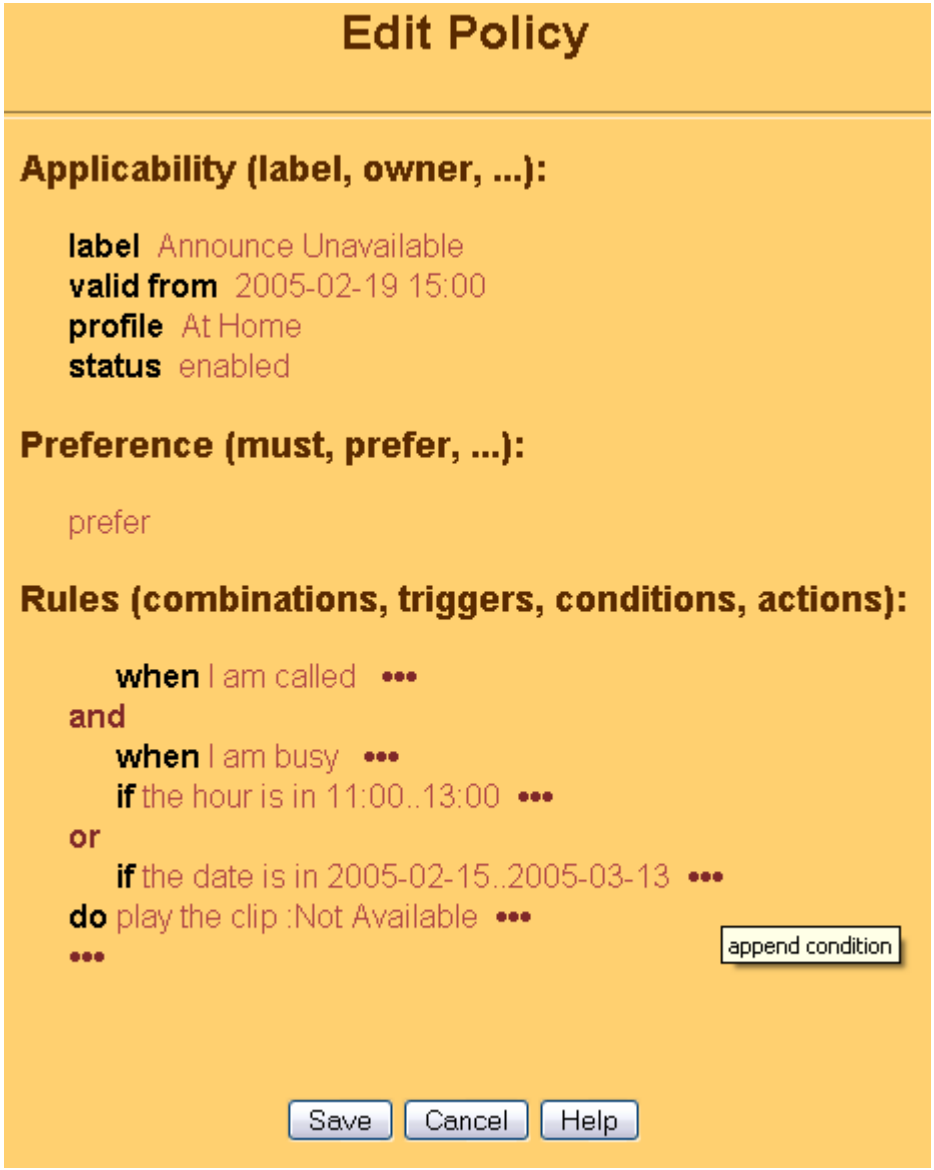


Figure 2.9: Edit Policy

Chapter 3

Policy Wizard Internals

3.1 Integration with Other Tools

The policy wizard uses a set of Java Server Pages (JSP). It therefore requires a servlet container such as Apache Tomcat. The wizard has been tested with Tomcat versions 4.1.27 onwards. Tomcat requires an application context to be set up. In Tomcat 5.X, the file:

tomcat/conf/Catalina/localhost/accent.xml

might have contents:

<**Context** path=″/accent″ docBase=″*wizard installation directory*″
debug=″0″ reloadable=″true″ crossContext=″false″>
<**Logger** className=″org.apache.catalina.logger.FileLogger″
directory=″/WEB-INF/logs″ prefix=″wizard_″ timestamp=″true″/>
</**Context**>

where the *wizard installation directory* might, for example, be *C:/Program Files/Tomcat/webapps/accent/wizard*.

The JSP is supplemented by Java code in package *uk.ac.stir.cs.accent.wizard*. Audio clips are handled by an adaptation of public domain code (http://dannyayers.com/2000/sound.htm).

To use the policy wizard requires a web browser with a recent version of HTML, Cascading Style Sheets (CSS) and JavaScript. The wizard has been tested with Microsoft Internet Explorer 6.0, Netscape Navigator 7.1 and Opera 7.5. The web browser must be JavaScript-enabled (for checking and form-handling) and also Java-enabled (if audio clips are to be used).

Multiple logins from the same web browser on the same client system are discouraged. As a result of browser/JSP limitations, it may result in the same session being used in all windows. In the current version of the wizard, all concurrent users must share the same interface language (UK English, Canadian French, etc.).

If a user wishes to record audio clips using the policy wizard, the relevant Java security policy must be set. This can be done with the command-line *policytool* provided with the Sun JDK. Alternatively, the file .java.policy in the user's home directory can be edited directly. It should contain an entry like the following:

grant codeBase ″http://*wizard server*/accent/wizard″ {
permission javax.sound.sampled.AudioPermission ″record″;
};

where the *wizard server* might be *www.stir.ac.uk:8080*. If it is an acceptable security risk, recording can be permitted for applets from any system by omitting the *codeBase* parameter.

The policy wizard requires a table called *users* within the *accent* database of the policy database server. This table must contain at least an entry for the policy wizard administrator (conventionally named *admin*). Several users may be designated as administrators. The sample file *lib/user_setup.sql* is provided as an example for this setup.

The policy wizard administrator requires an individual email address (e.g. admin@cs.stir.ac.uk) separate from any regular email address. The administrator creates policy system users, who then log in under their allocated username and password. These users would normally be from the same domain (e.g. cs.stir.ac.uk), but in principle could be from multiple domains.

3.2 Configuration

The policy wizard uses a Java properties file *database.properties* to obtain information about the policy database and the policy server. A typical configuration file looks like the following:

```
# General system administration

admin.email             kjt@cs.stir.ac.uk

# Database for user information

users.host              dubloon.cs.stir.ac.uk
users.password          password
users.table             accent
users.username          accent

# Policy server

policy.host             dubloon.cs.stir.ac.uk
policy.upload.port      9999
```

The policy wizard uses a Java properties file *mapping.properties* to map policy language terms to policy wizard terms. A typical configuration file looks like the following:

```
absent                  policy.absent
active_content          policy.active.content
add_medium              policy.add.medium
...

locale.de-de            language.de.de
locale.en-au            language.en.au
...

stage.0                 policy.novice
stage.1                 policy.intermediate
...
```

The policy wizard uses a Java properties file *wizard.properties* to obtain information about the mapping from policy wizard phrases to some natural language. A typical configuration file looks like the following:

```
# The following are interpreted by the browser and so may use HTML entities for
# special characters

aspect.applicability    Applicability (label, owner, ...)
aspect.preference       Preference (must, prefer, ...)
aspect.rule             Rules (combinations, triggers, conditions, actions)
...

# The following are interpreted by JavaScript and so cannot use HTML; escape
# a "'" character with "\\"

error.Address           Define address in "person@domain", telephone number,
  or ":variable" format
error.address           Define address in "person@domain" or ":variable" format
error.applies.to        Define "applies to" in "person@domain" format
error.audio             audio
...
```

Note that some properties intentionally differ in their capitalisation (like *error.Address*, *error.address*).

APPEL is translated into natural language in two steps. For example 'connect_incoming' is first converted into 'policy.connect.in' using the mapping properties. This is then translated into 'I am called' using the English wizard properties. A *Login* page is defined for each language in the *wizard* directory. One of these should be selected as the default for the locale, and linked or copied to *index.jsp*.

Element	Novice	Intermediate	Expert
Modality	must, must_not, prefer, prefer_not, should, should_not		
Combinator	*condition*: and, or *trigger*: else, or, sequential		*trigger*: and, andthen, guarded, orelse, parallel, unguarded
Trigger	absent(), available(), connect, connect_incoming, connect_outgoing, disconnect, present(), unavailable()	absent(address), available(address), disconnect_incoming, disconnect_outgoing, no_answer(period), no_answer_incoming(period), no_answer_outgoing(period), present(address), unavailable(address)	bandwidth_request, event, register, register_incoming, register_outgoing
Condition	caller, date, time, topic	call_type, cost	active_content, bandwidth, call_content, callee, capability, capability_set, destination_address, device, location, medium, network_type, priority, quality, role, signalling_address, source_address, traffic_load
Action	forward_to(Address), note_availability(true), note_availability(false), note_presence(true), note_presence(false), play_clip(audio), reject_call(reason), send_message(address,message)	log_event(message), note_availability(topic), note_presence(location)	add_caller(method), add_medium(medium), add_party(Address), confirm_bandwidth, connect_to(Address), fork_to(Address), reject_bandwidth(limit), remove_medium(medium), remove_party(Address)

Figure 3.1: Language Levels

3.3 Language Levels

The policy wizard restricts the use of certain language features depending on the user's defined level (called 'stage' internally). This is summarised in figure 3.1; see [4] for the meaning of these language terms. The columns are cumulative, e.g. an intermediate user can do everything a novice can plus whatever is listed in this column. An administrator is the highest level. The only difference from expert level is that an administrator can see and alter the *owner* and *applies_to* fields, i.e. can define policies and policy variables for others. An administrator can also, of course, manage users.

3.4 Templates

Template policies are XML files conforming to the APPEL schema, except that *owner* and *applies_to* are meaningless and are left empty. The files *must* be validated outside the policy wizard. Template policies exist in each locale directory (e.g. 'en-ca'). In principle they could be similar for each locale, though it is possible to have a different set according to local custom. The *id* values must be translated into the relevant language. In addition, the names of template variables must be rendered in the relevant language (e.g. '?address' in English, '?adresse' in

French). If the language requires special characters (e.g. accented ones), be sure to save a template file in UTF-8 format.

Template policies are suffixed by the user level for which they are intended ('_0.xml' novice, '_1.xml' intermediate, '_2.xml' expert). Template policies are sorted by filename, though their *id* is shown in the list. The filename should therefore closely match the *id*. Templates that should appear at the start of the list (e.g. the blank policy) should have their filenames prefixed by '0'. Only templates appropriate to the user level are shown when 'From Template' is selected.

3.5 Code Organisation

The code is organised in the following directories. The structure conforms to the normal one for Tomcat.

.classpath, .project: Eclipse build files

clip: the *Clip* applet for recording and playing audio clips

wizard: the JSP files, the CSS stylesheet, the wizard logo, and a JAR file for the *Clip* classes

WEB-INF: all the supporting files

WEB-INF/classes: the compiled Java files in package *uk/ac/stir/cs/wizard*

WEB-INF/doc: JavaDoc for the Java source files

WEB-INF/lib: the database properties file, the mapping properties file, the mySQL connector jar file, and language directories

WEB-INF/lib/*language*: the wizard properties file for some language (e.g. *en-gb*, meaning 'English – Great Britain'), the pre-defined policy templates for this language

WEB-INF/logs: Tomcat log files

WEB-INF/src: the Java source files in package *uk/ac/stir/cs/wizard*; there is a simple *build* script to recompile everything

WEB-INF/web.xml: the Tomcat web deployment descriptor

3.6 Internationalisation

The policy wizard is designed to be multi-lingual. Suppose that it is required to add Swiss German ('de-ch') to the list of supported languages. The steps required are as follows. If the language requires special characters (e.g. accented ones), be sure to save the files in UTF-8 format.

- Create the directory *WEB-INF/lib/de-ch*. This must contain the file *wizard.properties* and template policies for this language. If this is a variant on an already supported language (e.g. 'de-de'), the files from this can be copied and adjusted. If this is a completely new language, all terms required by the policy wizard will have to be translated. This can be tricky if the result is to be grammatically correct (e.g. nouns, adjectives and verbs agree). Judicious choices in the translation can make this possible for many (though not all) languages. The templates pre-defined for English can mainly be copied as they are. The main change required is to render the comments, the policy identifiers and the policy variables in the new language.
- Create the file *wizard/Login-de-ch.jsp*. This will be almost identical to the existing login files, but a couple of phrases need to be rendered in the new language. The *locale* also needs to be set as 'de-ch'.
- Create the file *wizard/Help-de-ch.jsp*. This may be a variation on an existing language help file (*Help-de-de.jsp* for example), or a completely new translation of it.
- Edit the mapping properties file to add the new locale 'de-ch' and its equivalent policy wizard term:

```
locale.de-ch            language.de.ch
```

Note the use of '-' in locales but '.' in policy wizard terms.

- In the wizard properties file for *every* language, define the translation of 'language.de.ch'. For example, in English this will be 'German - Switzerland', in French 'Allemand - Suisse', and in German 'Deutsch - Schweiz'.

References

[1] Stephan Reiff-Marganiec and Kenneth J. Turner. Use of logic to describe enhanced communications services. In Doron A. Peled and Moshe Y. Vardi, editors, *Proc. Formal Techniques for Networked and Distributed Systems (FORTE XV)*, number 2529 in Lecture Notes in Computer Science, pages 130–145. Springer, Berlin, Germany, November 2002.

[2] Stephan Reiff-Marganiec and Kenneth J. Turner. A policy architecture for enhancing and controlling features. In Daniel Amyot and Luigi Logrippo, editors, *Proc. 7th. Feature Interactions in Telecommunications and Software Systems*, pages 239–246. IOS Press, Amsterdam, Netherlands, June 2003.

[3] Stephan Reiff-Marganiec and Kenneth J. Turner. The ACCENT policy server. Technical Report CSM-164, Department of Computing Science and Mathematics, University of Stirling, UK, December 2005.

[4] Stephan Reiff-Marganiec, Kenneth J. Turner, and Lynne Blair. APPEL: The ACCENT project policy environment/language. Technical Report CSM-161, Department of Computing Science and Mathematics, University of Stirling, UK, December 2005.