Department of Computing Science and Mathematics
University of Stirling

The ACCENT Policy Wizard

Kenneth J. Turner and Gavin A. Campbell

Technical Report CSM-166

ISSN 1460-9673

April 2009

Department of Computing Science and Mathematics
University of Stirling

The ACCENT Policy Wizard

Kenneth J. Turner and Gavin A. Campbell

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

Telephone +44 1786 467 421, Facsimile +44 1786 464 551
Email kjt@cs.stir.ac.uk, gca@cs.stir.ac.uk

Technical Report CSM-166

ISSN 1460-9673

April 2009

Abstract

The ACCENT project (Advanced Component Control Enhancing Network Technologies) developed a practical and comprehensive policy system for call control/Internet telephony. The policy system has subsequently been extended for management of sensor networks/wind farms and of home care/telecare.

This report focuses on a web-based policy wizard that acts as the primary interface between end users and the policy system. The policy wizard has an intimate knowledge of the APPEL policy language (Adaptable and Programmable Policy Environment and Language). The wizard allows end users to create policies using near-natural language without knowing or seeing XML, and to upload them to the policy system. The wizard also provides a number of convenience functions such as predefined policy templates, editing and activating existing policies, and defining policy variables.

Relative to the version of December 2005, this Technical Report has been updated as follows to reflect changes in the policy wizard:

- The whole report has been updated to reflect later work on the PROSEN and MATCH projects. As a result, the ACCENT and APPEL acronyms have changed. Call control, however, remains are the primary illustration of the approach in this report.
- Chapter 1 is now named 'Introduction', and a brief 'Conclusion' chapter has been added in section 4.
- Chapter 2 has been updated to sheet screenshots of the new policy wizard. The wizard now handles resolution policies.
- The wizard now makes use of ontologies, as described briefly in section 3.1. All domain-specific knowledge is held outside the wizard, so that largely common code can be used across all domains. As a result, the wizard configuration now also refers to the POPPET server.
- Section 3.6 describes a new code structure that allows different versions of the wizard to coexist.
- A brief explanation has been given in section 3.8 of what is involved in supporting a new application domain with the wizard.

Contents

List of Figures

Chapter 1

Introduction

The ACCENT project (Advanced Component Control Enhancing Network Technologies, http://www.cs.stir.ac.uk/accent) developed a practical and comprehensive policy system for call control/Internet telephony. The policy system has subsequently been extended for management of sensor networks/wind farms on the PROSEN project (Proactive Control of Sensor Networks, http://www.prosen.org.uk). The policy system has also been extended for management of home care/telecare on the MATCH project (Mobilising Advanced Technologies for Care at Home, http://www.match-project.org.uk).

This report focuses on a web-based policy wizard that acts as the primary interface between end users and the policy system. The policy wizard has an intimate knowledge of the APPEL policy language (Adaptable and Programmable Policy Environment and Language, http://www.cs.stir.ac.uk/appel). The wizard makes use of domain-specific ontologies so that it can be used in any application. The wizard allows end users to create policies using near-natural language without knowing or seeing XML, and to upload them to the policy system. The wizard also provides a number of convenience functions such as predefined policy templates, editing and activating existing policies, and defining policy variables. Besides regular policies, the wizard supports resolution policies for handling conflicts among policy actions.

[4, 5, 8] give some general background to the ACCENT project. There are technical reports describing the ACCENT Policy Server ([6]) and the APPEL Policy Language ([7]). Ontologies for the policy wizard are discussed in [1, 2, 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 creating 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. For the wizard described in this report, 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. Other wizards using voice-based input (VoiceXML) and digital pen-based input (Anoto, Logitech) have been prototyped.
- The wizard is multilingual (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.

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 wizard is domain-independent. However, for illustration the following screenshots focus on its use for call control/Internet telephony. The operation is broadly similar in other application domains such as home care/telecare and sensor networks/wind farms. The screenshots show an administrator login as only this category of user sees all of the wizard features.

- The login screen in figure 2.1 is straightforward.
- The main menu in figure 2.2 shows 'Edit Users' and 'Edit Resolution' 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. Resolution policies can also be edited in a similar fashion.
- Choosing 'From Template' leads to figure 2.4, where a number of predefined policies can be selected and edited. Only administrators see templates for resolution policies. 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.

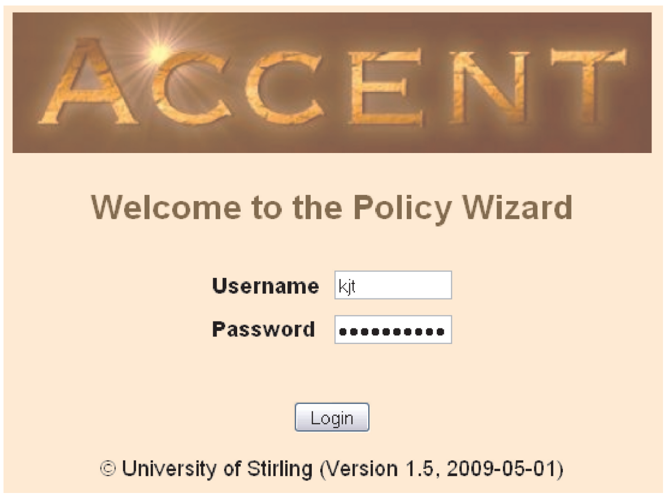

ACCENT
Welcome to the Policy Wizard
Username
kjt
Password
Login
© University of Stirling (Version 1.5, 2009-05-01)

Figure 2.1: Login Screen

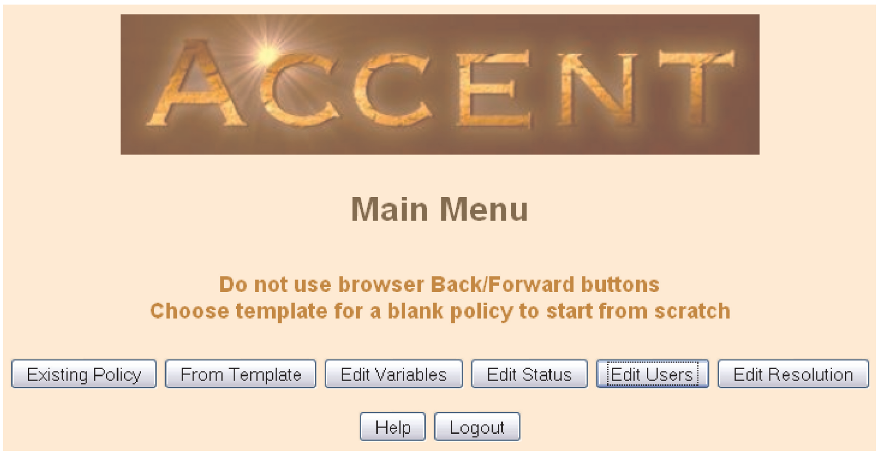


Figure 2.2: Main Menu

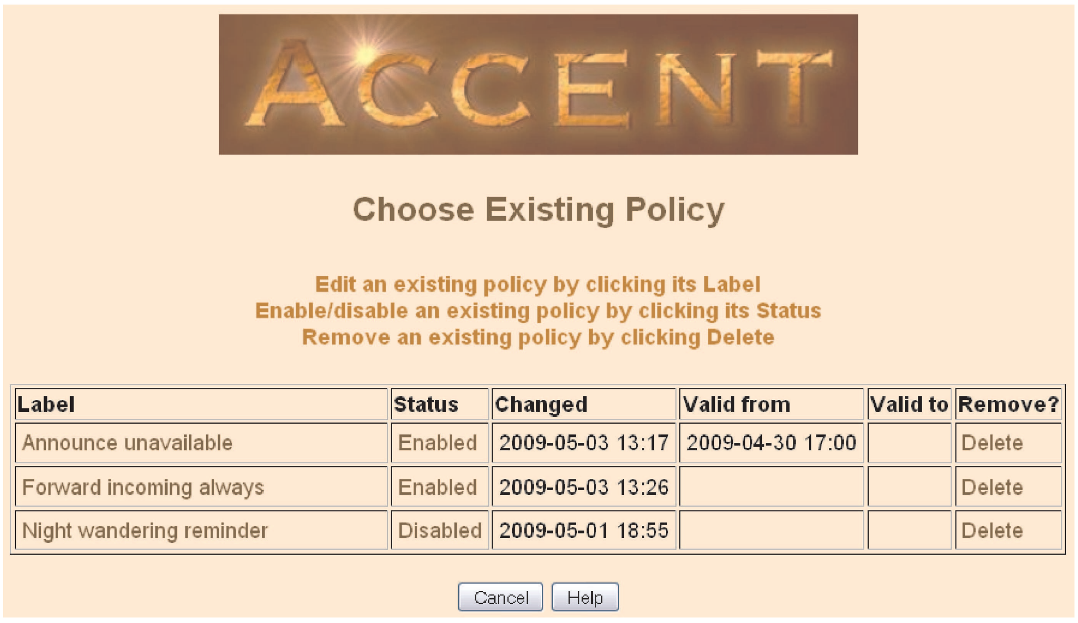
- Choosing 'Edit Variables' leads to figure 2.5, where an existing variable can be selected for editing or deletion. Variables are created normally created in text format.

- For use in voice application domains, audio clips are also supported as special kinds of variable values: WAV format, i.e. 16-bit uncompressed PCM (Pulse Code Modulation). Editing an audio clip is illustrated in figure 2.6. This provides buttons for record (red circle), stop (blue square) and play (green triangle). The meter bar shows the audio level during recording or the progress through the clip while playing.

- For use in voice application domains, 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.

- An administrator can choose'Edit Users', leading 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 '· · ·' symbol after a trigger, condition or action allows a further trigger, condition or



Choose Existing Policy

Edit an existing policy by clicking its Label
Enable/disable an existing policy by clicking its Status
Remove an existing policy by clicking Delete

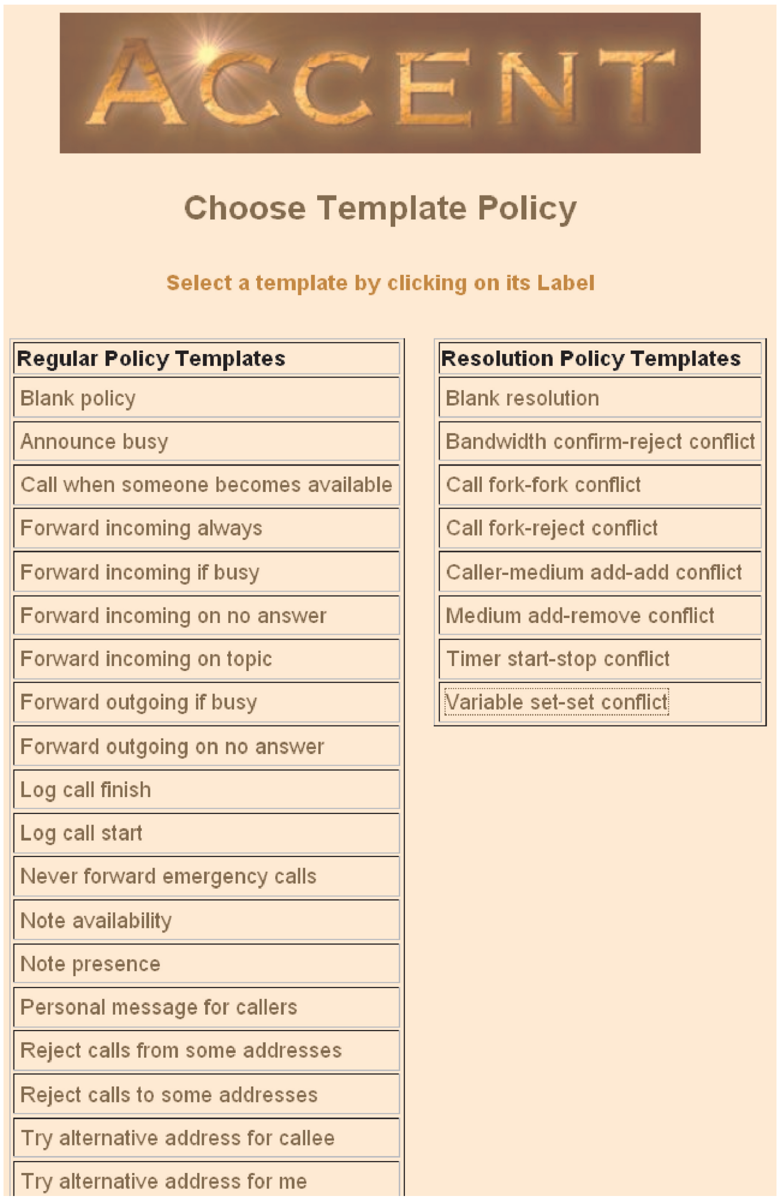
Label	Status	Changed	Valid from	Valid to	Remove?
Announce unavailable	Enabled	2009-05-03 13:17	2009-04-30 17:00		Delete
Forward incoming always	Enabled	2009-05-03 13:26			Delete
Night wandering reminder	Disabled	2009-05-01 18:55			Delete

Figure 2.3: Existing Policies

action to be added with a specified combination. The '···' 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 '···' symbol after the first condition; this may be clicked to select the action.

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.

- An administrator can also edit a resolution policy. This is broadly similar to editing a regular policy, as shown in figure 2.10. The most obvious difference is that *preference* and *variable* values are mostly used in place of concrete values.



ACCENT

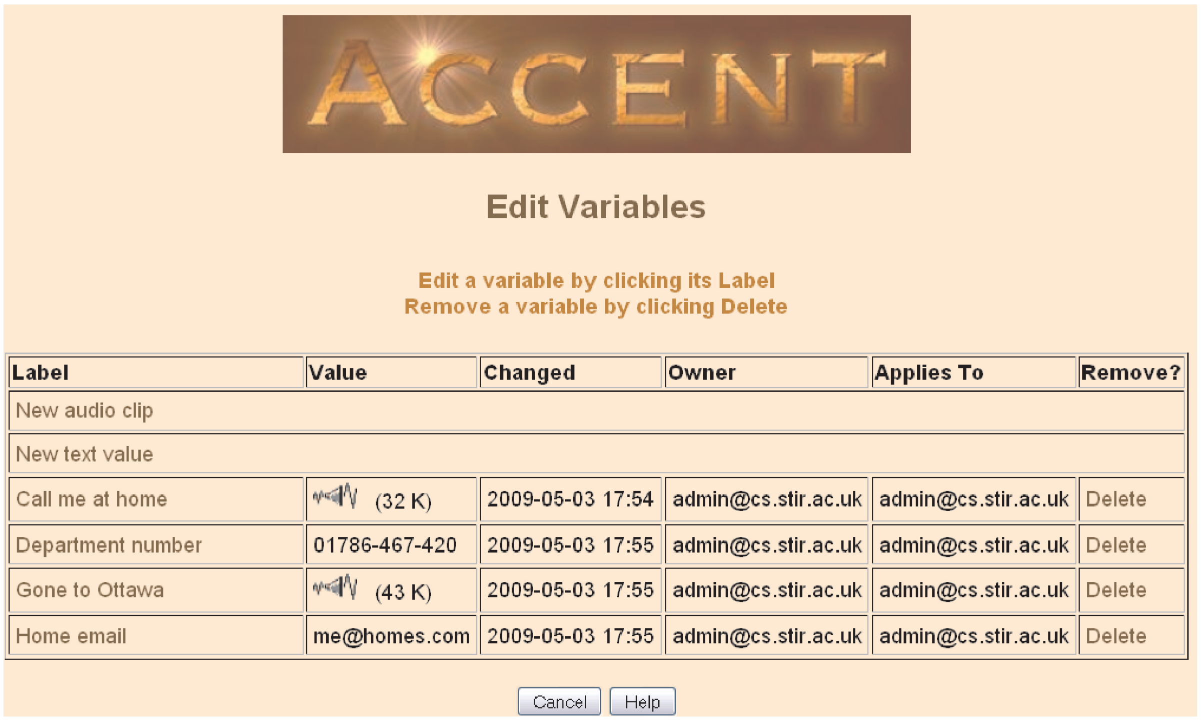
Choose Template Policy

Select a template by clicking on its Label

Regular Policy Templates
Blank policy
Announce busy
Call when someone becomes available
Forward incoming always
Forward incoming if busy
Forward incoming on no answer
Forward incoming on topic
Forward outgoing if busy
Forward outgoing on no answer
Log call finish
Log call start
Never forward emergency calls
Note availability
Note presence
Personal message for callers
Reject calls from some addresses
Reject calls to some addresses
Try alternative address for callee
Try alternative address for me

Resolution Policy Templates
Blank resolution
Bandwidth confirm-reject conflict
Call fork-fork conflict
Call fork-reject conflict
Caller-medium add-add conflict
Medium add-remove conflict
Timer start-stop conflict
Variable set-set conflict

Figure 2.4: Template Policies

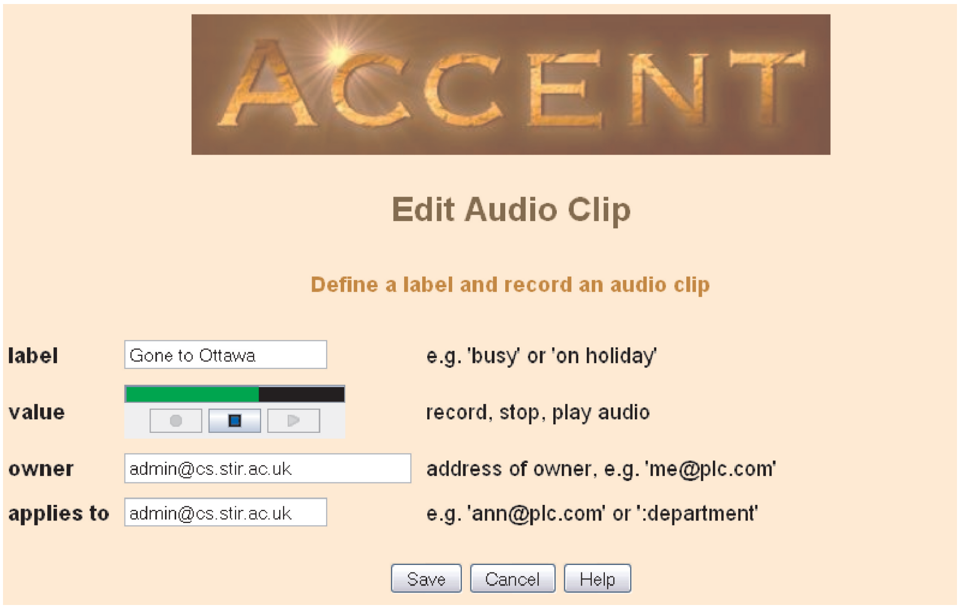


Edit Variables

Edit a variable by clicking its Label
Remove a variable by clicking Delete

Label	Value	Changed	Owner	Applies To	Remove?
New audio clip					
New text value					
Call me at home	(32 K)	2009-05-03 17:54	admin@cs.stir.ac.uk	admin@cs.stir.ac.uk	Delete
Department number	01786-467-420	2009-05-03 17:55	admin@cs.stir.ac.uk	admin@cs.stir.ac.uk	Delete
Gone to Ottawa	(43 K)	2009-05-03 17:55	admin@cs.stir.ac.uk	admin@cs.stir.ac.uk	Delete
Home email	me@homes.com	2009-05-03 17:55	admin@cs.stir.ac.uk	admin@cs.stir.ac.uk	Delete

Figure 2.5: Edit Variables

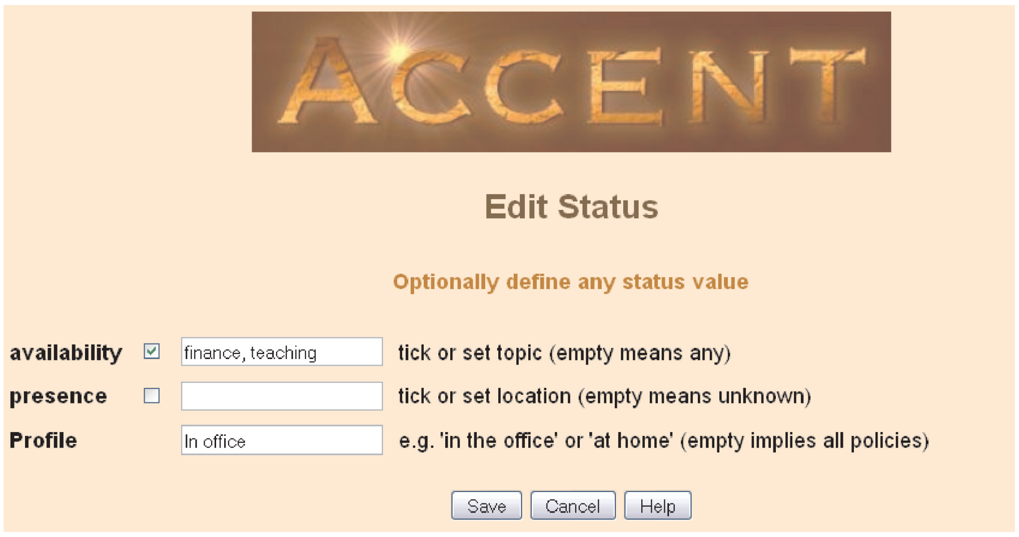


Edit Audio Clip

Define a label and record an audio clip

label	Gone to Ottawa	e.g. 'busy' or 'on holiday'
value		record, stop, play audio
owner	admin@cs.stir.ac.uk	address of owner, e.g. 'me@plc.com'
applies to	admin@cs.stir.ac.uk	e.g. 'ann@plc.com' or ':department'

Figure 2.6: Edit Audio Clip

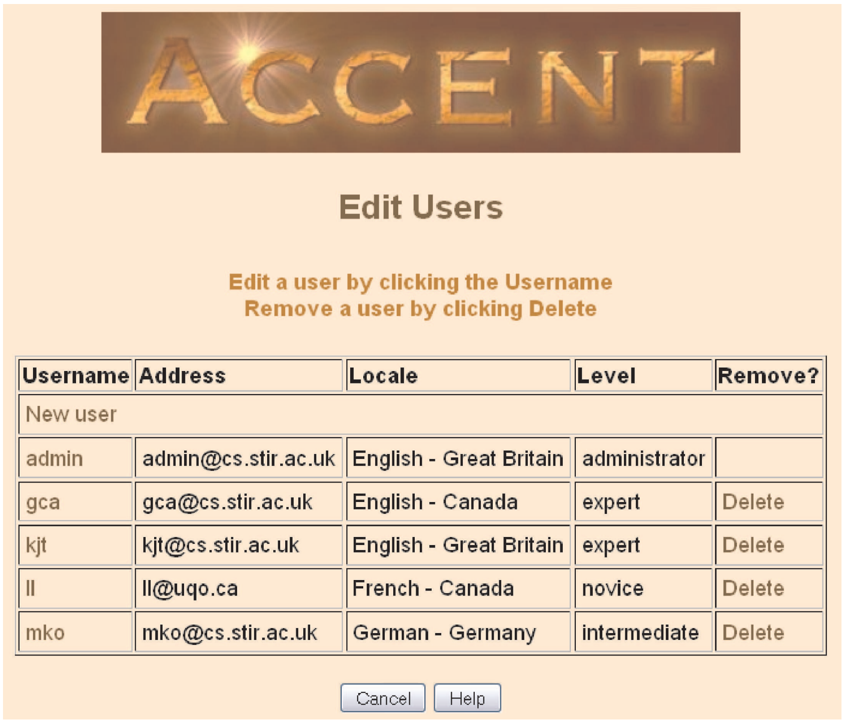


Edit Status

Optionally define any status value

availability	✓	finance, teaching	tick or set topic (empty means any)
presence	☐		tick or set location (empty means unknown)
Profile		In office	e.g. 'in the office' or 'at home' (empty implies all policies)

Figure 2.7: Edit Status

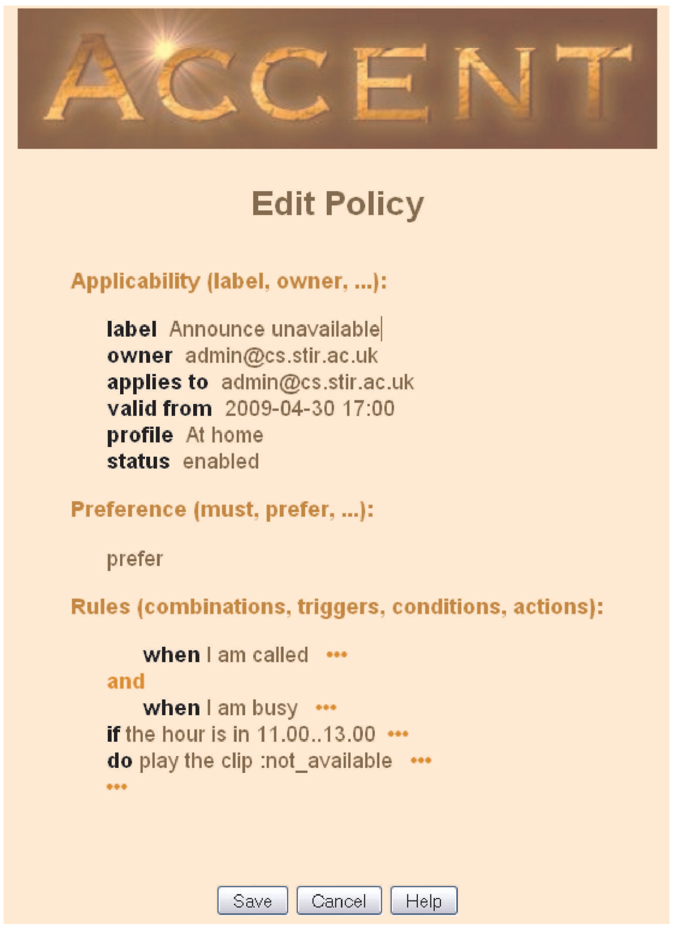


Edit Users

Edit a user by clicking the Username
Remove a user by clicking Delete

Username	Address	Locale	Level	Remove?
New user				
admin	admin@cs.stir.ac.uk	English - Great Britain	administrator	
gca	gca@cs.stir.ac.uk	English - Canada	expert	Delete
kjt	kjt@cs.stir.ac.uk	English - Great Britain	expert	Delete
ll	ll@uqo.ca	French - Canada	novice	Delete
mko	mko@cs.stir.ac.uk	German - Germany	intermediate	Delete

Figure 2.8: Edit Users



Edit Policy

Applicability (label, owner, ...):

label Announce unavailable
owner admin@cs.stir.ac.uk
applies to admin@cs.stir.ac.uk
valid from 2009-04-30 17:00
profile At home
status enabled

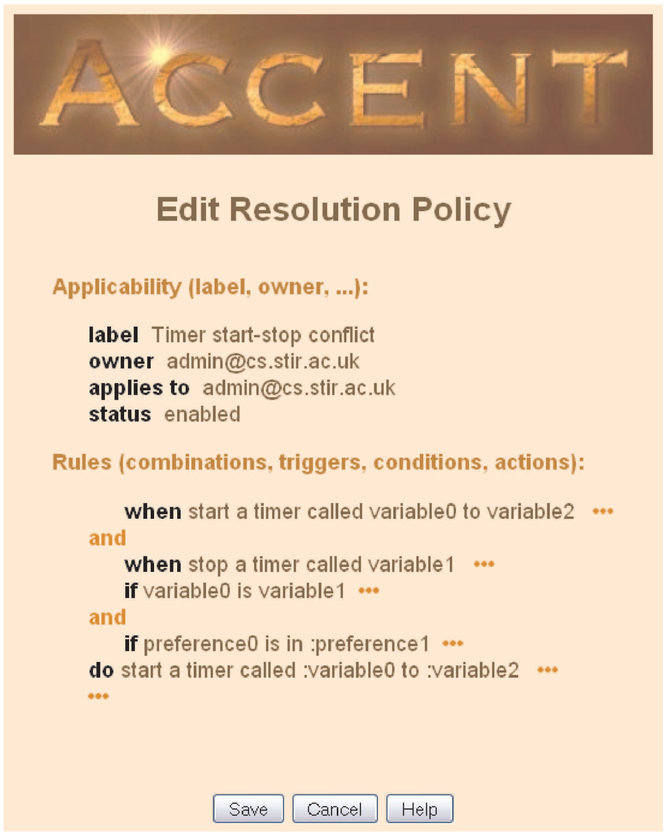
Preference (must, prefer, ...):

prefer

Rules (combinations, triggers, conditions, actions):

```
    when I am called •••
and
    when I am busy •••
if the hour is in 11.00..13.00 •••
do play the clip :not_available •••
•••
```

Figure 2.9: Edit Policy



Edit Resolution Policy

Applicability (label, owner, ...):

label Timer start-stop conflict
owner admin@cs.stir.ac.uk
applies to admin@cs.stir.ac.uk
status enabled

Rules (combinations, triggers, conditions, actions):

```
    when start a timer called variable0 to variable2 •••
and
    when stop a timer called variable1 •••
    if variable0 is variable1 •••
and
    if preference0 is in :preference1 •••
do start a timer called :variable0 to :variable2 •••
•••
```

Figure 2.10: Edit Resolution

Chapter 3

Policy Wizard Internals

3.1 Use of Ontologies

The wizard is designed to be used in multiple domains, and so obtains domain-specific information from a separate ontology server. The operation of the wizard is completely driven by this. POPPET (Policy Ontology Parser Program Extensible Translation [3]) provides an neutral interface for retrieval of information held in ontologies. Specifically, POPPET supports programmatic access to ontologies defined using OWL (Web Ontology Language [10]). Sample ontologies are described in [1, 2]. In fact a number of ontologies are used:

genpol: This is the generic policy ontology that applies to policies in any domain.

wizpol: This is the wizard policy ontology that contains additional information about how the wizard operates.

call_control, etc.: These are domain-specific ontologies that contain information about domain triggers, conditions, actions, units, user levels, etc.

When POPPET is instantiated for a particularly ontology, it builds a model of the ontology to support queries from external programs. This is performed through Java RMI (Remote Method Invocation), which enables the wizard to access the ontology model remotely. POPPET can be instantiated multiple times on the same system to support a variety of ontologies concurrently.

POPPET must be started before any attempt is made to use the wizard. As discussed below, the wizard has a configuration file that defines the POPPET server and ontology in use. POPPET is managed from the command line on the desired system using scripts provided in the bin directory of the POPPET distribution:

- The Bourne shell script po-start is provided to start POPPET. Call this with the command-line option '-h' (help) for more details. If the Java RMI registry is not already running, it is started. The po-start command-line option '-o' (ontology) defines the ontology to be used, e.g. 'c' (call control), 'h' (home care), 's' (sensor network).
- The Bourne shell script po-stop takes the same '-o' option and stops the POPPET server. This makes use of the PID (Process Identifier) stored in a file such as po_c.pid. If the Java RMI registry is running, it is also stopped.
- A much cruder Windows batch script po-start.bat is provided for those without a Unix shell.

3.2 Integration with Other Tools

The policy wizard uses a set of JSPs (Java Server Pages). 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 is Tomcat/webapps/accent/wizard, for example.

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 onwards, FireFox 2.0 onwards, Opera 7.5 onwards, and Safari 3.1 onwards. 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. With Internet Eplorer, it is possible to create a new session to avoid this. Although the policy wizard supports multiple domains, it is undesirable to do this with the same policy server. Policies and variables created in different domains could then all be visible to the policy server and wizard.

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.

3.3 Configuration

The policy wizard uses a Java properties file database.properties to define information about various servers. A typical configuration file looks like the following; substitute locally meaningful values here. Note that ontology URLs must end with '#'.

```
# General system administration

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

# Database for user information

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

# Policy server

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

# Poppet Ontology Server

poppet.host             poppet.cs.stir.ac.uk
poppet.ontology.name    call_control
```

```
# Ontology Locations

ontology.policy.generic     http://www.cs.stir.ac.uk/schemas/genpol.owl#
ontology.policy.wizard      http://www.cs.stir.ac.uk/schemas/wizpol.owl#
ontology.policy.domain      http://www.cs.stir.ac.uk/schemas/call_control.owl#
```

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
...
```

For each natural language, the policy wizard uses a Java property file wizard.properties to obtain information about the mapping from policy wizard phrases to the 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 (say) ‘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.

3.4 Language Levels

The policy wizard restricts the use of certain language features depending on the user’s defined level (called ‘stage’ internally). These levels are domain-specific, and so are defined by the domain ontology. As an example, the levels for call control are shown in figure 3.1. [7] describes 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 other users.

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 for Call Control

3.5 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 regular policies are suffixed by the user level for which they are intended ('_ 0.xml' novice, '_ 1.xml' intermediate, '_ 2.xml' expert). Template resolution policies are suffixed by '_res.xml'. 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.

3.6 Code Organisation

The code is organised in the following directories, conforming to the normal structure for Tomcat but split up for each application domains.

.classpath, .project: Eclipse build files

wizard: sub-directories with JSP files, CSS stylesheet, wizard logo, and JAR file for audio clip classes (voice application domains)

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 in the clip and wizard sub-directories

WEB-INF/lib: the database properties file, the mapping properties file, the MySQL connector JAR file, the POPPET JAR file, and application domain sub-directories

WEB-INF/logs: Tomcat log files

WEB-INF/src: the Java source files in sub-directories uk/ac/stir/cs/clip (audio clips) and uk/ac/stir/cs/wizard (policy wizard); there is a simple build script to recompile everything

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

The wizard and lib directories are subdivided according to the application domain: call_control, home_care and sensor_ network currently. The application domains under lib also also subdivided according to language. For example, 'en-GB' ('English – Great Britain') contains the wizard properties file and predefined policy templates for this language.

3.7 Internationalisation

The policy wizard is designed to be multilingual. 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 UTF8 format.

- Create the sub-directory de-CH in the relevant application domain sub-directory of lib. 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 predefined 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 Login-de-CH.jsp in the relevant application domain sub-directory of wizard. 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 Help-de-CH.jsp in the relevant application domain sub-directory of wizard. This may be a variation on an existing language help file (Help-de-DE.jsp for example), or a completely new translation.
- Edit the mapping properties file in the lib directory 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* application domain and 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'.

3.8 New Domains

Extending the wizard for other application domains is a much more substantial exercise. The code of the wizard does not need to be altered. But following need to be defined:

- An ontology for the new domain will be required. This might be based on one of the existing examples such as call_control.owl. An ontology editor such as PROTÉGÉ (http://protege.stanford.edu) is suggested for this.
- The database.properties configuration file must refer to the ontology created for the new domain.
- The mapping.properties and wizard.properties configuration files must be edited to refer to relevant terms in the new domain.
- The JSPs must be modified to reflect the new domain. The existing files in the wizard directory should serve as suitable examples of what is needed.

Chapter 4

Conclusion

The policy wizard described in this report has been successful in allowing non-technical users to defines policies. Being web-based, it can be used anywhere; for example, a user can remotely modify policies for handling calls. By using near-natural language, the wizard can support users in their local language. Other conveniences such as template policies and policy variables make it possible for an administrator to customise the wizard for local usage.

The use of domain-specific ontologies allows the policy wizard to operate in a wide variety of applications. Currently, the policy wizard (and system) have been used to manage call control/Internet telephony, sensor networks/wind farms, and home care/telecare. It is believed that the approach is generic and can be applied in many other management applications.

Two other experimental versions of the policy wizard have been created, though they are not described here. A second wizard makes use of VoiceXML [9] to allow policy contents to be read out, and template policies to be instantiated with specific parameters. A third wizard makes use of digital pen and paper (from Anoto or Logitech) to allow template policies to be completed through filling in paper forms.

References

[1] Gavin A. Campbell. Ontology for call control. Technical Report CSM-170, Department of Computing Science and Mathematics, University of Stirling, UK, June 2006.

[2] Gavin A. Campbell. Ontology stack for a policy wizard. Technical Report CSM-169, Department of Computing Science and Mathematics, University of Stirling, UK, June 2006.

[3] Gavin A. Campbell. Overview of policy-based management using POPPET. Technical Report CSM-168, Department of Computing Science and Mathematics, University of Stirling, UK, June 2006.

[4] 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.

[5] 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.

[6] Stephan Reiff-Marganiec, Kenneth J. Turner, and Feng Wang. The ACCENT policy server. Technical Report CSM-164, Department of Computing Science and Mathematics, University of Stirling, UK, April 2009.

[7] Kenneth J. Turner, Stephan Reiff-Marganiec, Lynne Blair, Gavin A. Campbell, and Feng Wang. APPEL: The ACCENT project policy environment/language. Technical Report CSM-161, Department of Computing Science and Mathematics, University of Stirling, UK, April 2009.

[8] Kenneth J. Turner, Stephan Reiff-Marganiec, Lynne Blair, Jianxiong Pang, Tom Gray, Peter Perry, and Joe Ireland. Policy support for call control. *Computer Standards and Interfaces*, 28(6):635–649, June 2006.

[9] VoiceXML Forum. *Voice eXtensible Markup Language*. VoiceXML Version 2.0. VoiceXML Forum, January 2003.

[10] World Wide Web Consortium. *Web Ontology Language (OWL) – Reference*. Version 1.0. World Wide Web Consortium, Geneva, Switzerland, February 2004.