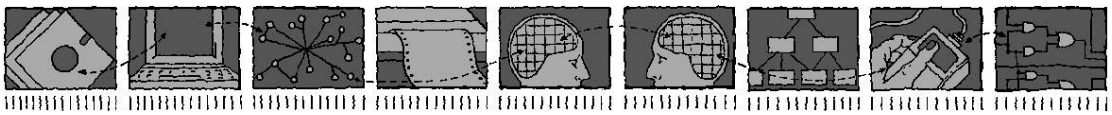
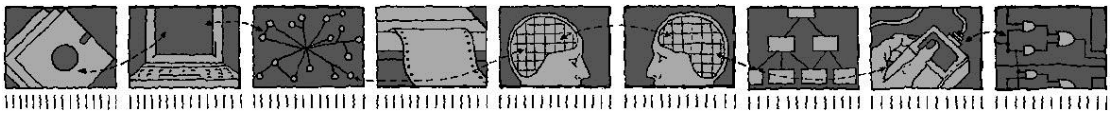


Computing Science and Mathematics
University of Stirling



The ACCENT Policy System for Home Care

Kenneth J. Turner

Technical Report CSM-188

ISSN 1460-9673

January 2017

Computing Science and Mathematics
University of Stirling

The ACCENT Policy System for Home Care

Kenneth J. Turner

Computing Science and Mathematics
University of Stirling
Stirling FK9 4LA, Scotland
Telephone +44-1786-467423, Facsimile +44-1786-464-551

Email kjt@cs.stir.ac.uk

Technical Report CSM-188

ISSN 1460-9673

January 2017

Abstract

This report describes the architecture, installation and configuration of the ACCENT policy system. It is seen that virtually all the ACCENT components are bundles deployed on an OSGi system. These bundles communicate using the OSGi event service. The details are given of how to set up and configure each of the bundles.

Keywords: ACCENT (Advanced Component Control Enhancing Network Technologies), APPEL policy language (ACCENT Project Policy Environment/Language), goal, OSGi (Open Systems Gateway initiative), policy.

Changes in Version 2

Relative to the version of April 2011:

- support for Plugwise energy monitors has been added in sections 2.10 and 3.4.17.

Changes in Version 3

Relative to the version of May 2011:

- barcode reader hardware is now introduced in section 2.1, the corresponding driver has been listed in section 3.4.2, and details have been given in section 3.4.5
- RFID reader hardware is now introduced in section 2.1, the corresponding driver has been listed in section 3.4.2, and details have been given in section 3.4.18
- an overview of Knopflerfish is now given in section 3.4.1, including an explanation of how to run Knopflerfish in the background as a Windows service
- the policy server parameters in section 3.4.15 have been slightly updated
- running the policy wizard in Tomcat is now described in section 3.4.16

Changes in Version 4

Relative to the version of September 2012:

- a description of the Raspberry Pi small computer that has been used to run Accent is now given in section 2.1
- use of Java and serial ports on the Raspberry Pi are now described in sections 3.1 and 3.2
- the position with regard to 32-bit and 64-bit operating systems is now discussed in section 3.1
- the audio and speech player is now mentioned in section 3.4.2, and details have been given in section 3.4.4
- observations about running the Ontology Server remotely have been added to section 3.4.13
- the location of Policy Wizard property files has changed as described in section 3.4.16

Changes in Version 5

Relative to the version of February 2013:

- section 3.4.1 explains that bundles are now configured by means of property files located according to the *uk.ac.stir.cs.accent* system property (they were previously placed in the Knopflerfish *osgi* top-level directory)
- the offline conflict analyser is now described in sections 3.4.2 and 3.4.7
- section 3.4.15 has been updated to reflect the current policy server properties (in particular, the server debug flags have changed slightly)
- in section 3.4.16, the *uk.ac.stir.cs.accent* property now determines whether the policy wizard is running as a bundle, and the policy wizard mapping file is now always internal

Changes in Version 6

Relative to the version of June 2013:

- the title of this report now qualifies its subject as home care
- the PolicyAction bundle has now been removed as IRTransDriver and X10Driver now directly perform *device_out* actions
- driver locations have now been mentioned in sections 2.2 and 2.3 for the ACR122U RFID reader and the Cipherlab 1070 barcode reader

- section 2.4 now describes the FitBit fitness monitor
- section 2.6 now describes the i-Buddy
- section 2.8 now describes the Nabaztag
- section 2.12 now describes Tunstall sensors
- section 2.13 now describes the TuxDroid
- section 3.1.2 now mentions the Java version 8 preview that works on the Raspberry Pi
- section 3.4.1 now mentions a Knopflerfish property to be set for unpacking bundle JARs
- section 3.4.2 now omits PolicyAction, with corresponding changes in start levels
- sections 3.4.2 now mentions the FitBitDriver start level
- sections 3.4.2 and 3.4.4 now mention that AudioPlayer starts at level 10
- sections 3.4.2 and 3.4.7 now describe the new ConfigurationSetup bundle
- section 3.4.4 now describes AudioPlayer changes for a new *audio.entity* property, speech synthesis to file, use of audio clips, and use of a preamble tone rather than a delay
- section 3.4.8 now defines an 'overwrite' parameter for the ConflictAnalyser bundle
- sections 3.4.2 and 3.4.9 describe the new FitBitDriver bundle
- sections 3.4.2 and 3.4.10 describe the new ForecastService bundle
- section 3.4.2 and 3.4.11 describe the new IBuddyDriver bundle
- section 3.4.12 now defines actions parameters for IRTransDriver, while section 3.4.26 now defines actions parameters for X10Driver
- sections 3.4.2 and 3.4.13 describe the new NabaztagDriver bundle
- section 3.4.14 now describes how ontologies can be served from the local host
- section 3.4.15 now defines the properties *variable.prefix* and *server.records* for the policy server, *server.log.lines* has been renamed *server.lines*, and a new 0400 debug flag has been added
- section 3.4.16 now defines the properties *policy.message.port*, *system.prefix*, *wizard.debug*, *device.actions* and *device.triggers* for the policy wizard
- section 3.4.22 describes the new SpeechRecogniser bundle
- sections 3.4.2 and 3.4.23 describe the new TunstallDriver bundle
- sections 3.4.2 and 3.4.25 describe the new TuxDroidDriver bundle

Changes in Version 7

Relative to the version of April 2014:

- section 3.4.10 describes the revised location parameter for ForecastService
- section 3.4.22 explains that SpeechRecogniser now conforms to the Google Speech API Version 2 and therefore requires a developer key parameter

Changes in Version 8

Relative to the version of July 2015:

- the overall framework now uses Knopflerfish 5.2.0, with consequent changes in some bundles
- section 3.1.1 no longer references *Jacspcsc.dll* and mentions that RFIDDriver will now work on 32-bit and 64-bit Windows
- FitBitDriver in section 3.4.9 has been updated for the latest FitBit API using OAuth 2.0; as a result, a number of properties have changed
- section 3.4.18 no longer references *Jacspcsc.dll* as the new RFIDDriver implementation uses the Java SmartCardIO API; the polling interval parameter is no longer used

Table of Contents

1 Introduction

This document describes the architecture, installation and configuration of the ACCENT policy system (Advanced Component Control Enhancing Network Technologies,). The policy system is comprehensive and flexible, with a variety of components developed on the projects ACCENT (http://www.cs.stir.ac.uk/accent), MATCH (http://www.match-project.org.uk) and PROSEN (http://www.cs.stir.ac.uk/~kjt/research/prosen). Among the major elements documented elsewhere are the APPEL policy language (ACCENT Project Policy Environment/Language [5], http://www.cs.stir.ac.uk/appel), the ACCENT Policy Server [2], the ACCENT Policy Wizard [4], and the ACCENT Ontology Server [1].

This report describes the major components of the policy system, how to install them, and how to configure them. The policy system supports a substantial amount of hardware and software. The main hardware used is as follows:

- Eston GPRS609 GPRS modem
- IRTrans modules for infrared control (USB and Ethernet variants)
- RFXCOM USB receivers for Oregon Scientific and Visonic sensors (433 + 867 MHz dual receiver or 867 MHz receiver)
- Oregon scientific wireless environment sensors
- Plugwise energy monitors
- Visonic wireless home sensors

The software versions below are those that have been used in testing, though later versions may be suitable:

- FireFox 3.N or Internet Explorer 8.N
- JavaComm 2.0 or SerialPort 8.2
- JAVE 1.0.2
- jNabServer 2.1 (with modifications)
- JRE 1.6
- Knopflerfish 5.2.0 with Xerces-J Bundle 2.10.1
- MySQL 5.0.67
- Pax Web Jetty Bundle 1.0.2, Pax Web JSP Support 1.0.2, Pax Web WAR Extender 0.8.0
- TSpaces 2.1.2

2 Hardware

2.1 Raspberry Pi

The Raspberry Pi (http://www.raspberrypi.org) is a small computer that runs Linux. It has been used to run the ACCENT system. The Raspberry Pi has an ARM v6 (ARM11) processor that runs at 700MHz, 256MB or 512MB memory, a 10/100Mbps Ethernet connection, two USB ports, an HDMI port, and other features. An SD/SDHC card is used as a solid-state disc. Although part of this is used as swap space, it is strongly preferable to have 512MB memory in order to minimise use of swapping. An Ethernet connection to the Internet is necessary for real time as the processor board does not have a real-time clock.

The recommended operating system is Raspbian, an optimised version of Debian 'Wheezy'. This comes in two variants: 'armel' that uses soft floating point, and 'armhf' that uses hard floating point. The former is preferable as it allows an Oracle Java 6 or 7 embedded JRE or OpenJDK Java 6 or 7 JRE to run. Java 8 embedded supports the hard floating point version of Raspbian.

Access to devices is controlled by rules under */etc/udev/rules.d*. USB devices that emulate a serial port are likely to be accessible by the ordinary user. However, other USB devices may be usable only by root. This can be fixed by creating a file such as *10-usb.rules* that places USB devices in group *staff* and makes them readable/writable by this group:

```
SUBSYSTEMS=="usb", GROUP="staff", MODE="0660"
```

2.2 ACS RFID Reader

The ACS (Advanced Card Systems) ACR122U has been tested with ACCENT. This requires a driver and JNI (Java Native Interface) code. The reader has been tested on Windows XP and Windows 7 32-bit/64-bit. The driver can be downloaded from:

http://www.acs.com.hk/index.php?pid=product&id=ACR122U

2.3 Cipherlab Barcode Reader

Many barcode readers emulate a keyboard so that barcodes appear as if typed. For use with the ACCENT system, a barcode reader needs to interface as a serial port. The Cipherlab 1070 has been tested with ACCENT. This can be set into VCOM (Virtual COM port) mode by scanning appropriate barcodes. The reader has been tested on Windows XP/Windows 7 32-bit and Windows 7 64-bit. The VCOM driver can be downloaded from:

http://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx

2.4 FitBit Fitness Monitor

FitBit fitness monitors such as the Flex are typically equipped with an accelerometer, a (simple) display, input by tapping the monitor, and vibration output. The monitors can be used to record lifestyle factors such as number of steps taken per day and sleep patterns. A dongle connects through Bluetooth to the wrist-band. With FitBit Connect running, the presence of a FitBit monitor in the area is checked every 15 minutes. The monitor has been tested on Windows XP and Windows 7 32-bit/64-bit. The FitBit Connect and FitBit Setup applications can be downloaded from a location such as:

https://www.fitbit.com/uk/setup

2.5 GPRS Modem

Any GPRS modem can, in principle be used. However, the code has been tested with an Eston GPRS609 (using a BenQ M32 chip, http://www.portech.com.tw/data/BenQ%20M23%20AT.pdf) that has a USB interface. The modem has been tested on Windows XP and Windows 7 32-bit (a 64-bit driver is not currently available).

2.6 i-Buddy Internet Buddy

The i-Buddy 'angel' is an Internet Buddy developed by Union Creations to signal information from Windows Messenger. The i-Buddy connects via USB. It has three head LEDs that can be set for different colours, a red heart light that can be turned on and off, wings that can be made to flap, and the ability to rotate left or right. The i-Buddy Manager is available in 32-bit and 64-bit versions and has been tested on Windows XP, Windows 7 and Windows 8. This program can be downloaded from:

http://www.snellelinks.nl/images/ibuddy/ibuddy210-f07.exe

The i-Buddy can also be controlled from Java using JLibiBuddy that can be downloaded from:

http://www.jraf.org/static/maven/sites/jlibibuddy

This requires the Java HID API available from:

http://code.google.com/p/javahidapi

The Java HID API DLL is available in 32-bit and 64-bit versions and has been tested on Windows XP, Windows 7 and Windows 8. This DLL (hidapi.dll) should be installed in \Windows\System32 or \Windows\SysWOW64 for 32-bit or 64-bit versions.

2.7 IRTrans Transmitter

An infra-red transmitter made by IRTrans (http://www.irtrans.de) is used to exchange infrared signals with domestic appliances such as TVs, CD players and DVD recorders. The transmitter has been tested on Raspberry Pi, Windows XP and Windows 7 32-bit/64-bit.

2.7.1 IRTrans Server

On Windows, the server *IRServer.exe* or *IRTransTray.exe* can be run from the IRTrans installation directory. On the Raspberry Pi, *irserver* should be run as root. In either case a parameter of USB should be used to find a USB-connected device.

After reading the definitions of remote controls (in subdirectory *remotes*), the server listens on port 21000 on the local host. To accept commands from another host, the firewall may need to be modified to accept connections on port 21000.

The commands for a remote control are identified in a *.rem* file in the *Remotes* sub-directory of the installation directory. The names of these files are converted to lower case (e.g. *TV_Lounge.rem* becomes the remote control *tv_lounge*).

A *.rem* file has a format such as:

```
[REMOTE]
  [NAME]TV_Lounge

[TIMING]
  [0][N]2[1]512 2024[2]512 4552[RC]3[RP]40[FREQ]39

[COMMANDS]
  [Vol+][T]0[D]0011101010010
```

This gives the *NAME* of the remote control and its *TIMING*. These are followed by the commands for the remote control such as *Vol+*. A file like this is created using the *Learn* command of the server or of the GUI client. Choose a name for the remote. Then, in turn, enter a name for each command and click the *Learn* button. Point the remote control at the IRTrans LEDs and click the corresponding button on the remote control. The IRTrans will flash as it learns the command.

Some commands from a remote control toggle a bit in the signal. These can be accommodated by adding *[TOGGLE][4][01]* at the end of the timing.

2.7.2 IRTrans GUI Client

A GUI client *IRRemote.exe* is provided to communicate with the server, normally on the local host port 21000 and using TCP. This uses the file *Remote.irm* in the installation directory to display the layout of a remote control such as:

```
[TV]
[FRMPIX]230,220
[LBL]10,200     [SIZE]300,30[TEXT]TV[FONT]14
[SLED]1

[POS]10,10      [SIZE]110,30[TEXT]Power    [REMOTE]TV_Lounge    [COMMAND]Off
```

This defines a *TV* device section. *FRMPIX* gives the x,y extent of the remote control panel in a window. *LBL* defines one or more labels that appear in this window. *SLED* defines the use of LEDs (1 = internal).

Then each button is defined on a line of its own. A *POS* field gives the x,y position of a button, while *SIZE* defines its x,y extent. *TEXT* defines the button label. *REMOTE* identifies the file of IR commands for a remote control. *COMMAND* defines the command in this file associated with the button.

2.7.3 IRTrans Command-Line Client

Code in C is provided for a command-line client. Compile *irclient.c* as follows (possibly using *gcc*):

```
cc -o irclient -O irclient.c
strip irclient
```

The command-line client can now be called as:

```
irclient host_identifier remote_name command_name
```

e.g.:

```
irclient triton TV_Ferguson 3
irclient 10.0.0.18 TV_Ferguson Off
```

The remote name and command name can be in upper or lower case.

2.7.4 IRTrans UDP Communication

A UDP socket can be created to send commands to the IRTrans server. Send data in the format:

```
SND remote_name,command_name,Pport_number
```

e.g.:

```
SND TV_Ferguson,Off,P21001
```

and expect a response such as *OK* or *ERR*. *SND* and *P* can be all lower or all upper case. The port number is the one used for the response to the client (21000 by default). Note that the response is not made to the port used by the client to send the message.

2.8 Nabaztag Internet Rabbit

The Nabaztag 'Internet rabbit' (http://en.wikipedia.org/wiki/Nabaztag) was created by Violet. The Nabaztag Tag is the second-generation device. It connects using WiFi to a configured Internet address (originally a Violet server). However, it can also be set up to use other servers including a local one. For input the device has a microphone, an RFID tag reader, and movable ears. For output the device has a loudspeaker, multiple coloured lights, and movable ears. The Nabaztag has been tested on and has been tested on Windows XP, Windows 7 (64-bit) and Windows 8 (64-bit).

2.9 Oregon Scientific Wireless Sensors

Wireless sensors made by Oregon Scientific (http://www.oregonscientific.com) are used to measure environmental variables such as temperature and humidity. The BTHR918 sensor measures indoor temperature, humidity and atmospheric pressure. The THGR228 sensor measures indoor or outdoor temperature and humidity. The THGR918 sensor measures outdoor temperature and humidity

2.10 Plugwise Energy Monitors

Modules from Plugwise (http://www.plugwise.com) are used to monitor electricity usage. These modules ('Circles') are connected in a ZigBee network to a USB receiver ('Stick') plugged into a computer. One module ('Circle+') is designated as the network controller, e.g. it is responsible for the clock. The receiver has been tested on Windows XP and Windows 7 32-bit/64-bit. The receiver uses an FTDI USB interface chip, with drivers available from:

http://www.ftdichip.com/Drivers/VCP.htm

Plugwise provide software ('Source') for configuring and controlling the network, and for reading electricity usage from the modules. It is not clear if it is strictly necessary to initially configure the network using the software, but it is probably wise to do so. Updates to module firmware are managed via this software.

2.11 RFXCOM Wireless Receiver

A wireless receiver made by RFXCOM (http://www.rfxcom.com) is used to read signals from Oregon Scientific and Visonic sensors. The receiver has been tested on Raspberry Pi, Windows XP , Windows 7 32-bit/64-bit. The receiver uses an FTDI FT2232R USB interface chip, with drivers available from:

http://www.ftdichip.com/Drivers/VCP.htm

When plugged in, the receiver appears as a 'USB Serial Port' in Windows Device Manager. Note the COM port number that is allocated to it.

This receiver comes with a number of utilities on a mini CD. RFReceiver.exe on this can be used to view data from a COM port. By default the receiver works at 4800 bps. Before using the receiver with the policy system, it is necessary to set it for 38400 bps. Do this by running RFReceiver.exe, connecting a receiver, and clicking on 'Toggle Baud Rate (don't use)' to make it work at 38400 bps in future. The current setting can be checked by seeing if RFReceiver.exe can successfully decode a signal at the expected bit rate.

2.12 Tunstall Sensors

Tunstall (http://www.tunstall.co.uk) provide a wide range of sensors and actuators to support telecare and telehealth. Example sensors include the following models. When a sensor fires, its sensor identifier is sent as a

two-digit ‘location’: a resident identifier (e.g. 01) or a room identifier (e.g. 41 for the kitchen). A two-character sensor code is also sent to indicate the kind of signal.

Model	Sensor Type	Battery	State	Code	Notes
41005/12-C0	bed/chair occupancy	4×AA	occupied	AZ	use port IP2 or IP3; use port IP4 and Palm emulator to program
			unoccupied	BA	
D4106009A	mattress pad				for use with bed/chair occupancy sensor
41005/25-A0	universal sensor	Tadiran SL-360P	opened	AQ	set DIP switches to 1110 0001 for door monitoring in this way
			closed	AR	
67005/60-A0	medication dispenser	2×AA	dose missed	CZ	signals if two hours late in taking medication
67005/89-A1	fast PIR	PP3	movement	BH	
67005/02-J1	pendant alarm		pressed	AA	
68005/01-A7	wrist alarm		pressed	AA	

Tunstall sensors connect wirelessly to a Lifeline Connect+ base unit. To register a sensor, hold the green button for five seconds to enter programming mode; the red button then flashes slowly. Then hold the green button for a further three seconds to enter registration mode; the red button then flashes quickly. At this point, activate the sensor to register it with the base unit. Finally, press the green button to return to normal operation.

The Lifeline Connect+ is programmed by using the PC Connect application supplied by Tunstall. The Lifeline Connect+ has been tested on Windows XP and Windows 7 32-bit/64-bit. This can be downloaded from:

http://uk.tunstall.com/solutions/lifeline-vi

PC Connect interfaces to the Lifeline Connect+ via a Tapit+. Depending on the model, this requires an RS232 or USB port on the PC. An RS232-USB converter may be required such as the FTDI UC232R. The ‘User’ model of the Tapit+ can be used for configuration and also for receiving sensor data; the ‘Schools Protection’ model can also be used for sensor data. PC Connect allows locations to be assigned to wireless sensors (along with many other functions).

Some devices such as the bed/chair occupancy sensor or the property exit sensor are programmed using the Sensor Tool application for the Palm or Palm Emulator. A Palm Emulator and this application (version 1 or 2) can be obtained from Tunstall. The Palm Emulator requires a COM1 or COM2 serial port (or an RS232-USB converter to emulate this). The application (TIM icon) can be used to program the bed/chair occupancy pressure threshold (also set the ADLife option). When using version 1 of the application, press the Menu button and select Phase 2/Occupancy Sensor 151 or Phase 3/Occupancy Sensor 171, for example. When using version 2 of the application, select Cable and then OK.

2.13 TuxDroid

The TuxDroid ‘penguin’ (http://en.wikipedia.org/wiki/Tux_Droid) was developed by Kysoh as a desktop companion. It connects wirelessly through a USB dongle. For input the TuxDroid supports head and wing clicks, light-level readings and audio recording. For output the TuxDroid supports movement of eyelids, beak and wings, turning eye lights on and off, rotation of the body (unless connected to a charger) and audio playback. The TuxDroid has been tested on Windows XP , Windows 7 64-bit and Windows 8 64-bit. The TuxDroid can be used by Java with the TuxDroid-Java package available from:

https://github.com/Cicatrice/tuxdroid-java

This requires libtuxdriver.dll which is part of the Tux Droid Software Suite. The DLL works on Windows XP, Windows 7 and Windows 8, but is available only in 32-bit form.

2.14 Visonic Wireless Sensors

Example sensors include the following models. In general, code 84 means normal, code 04 means alert, and code 0C means activated.

Model	Sensor Type	Battery	State	Code	Notes
MCT-211	wrist alarm	CR2025	pressed	0C	
MCT-241	pendant alarm	CR2	pressed	0C	
MCT-302T	magnetic contact	CR2	open	04	same transmitter as MCT-550
			closed	84	
MCT-425	smoke	PP3 Long Life	smoke	04	
			clear	84	

MCT-441	natural gas	CR123A	gas	04	mains-powered with 2-pin plug
			clear	84	
MCT-442	carbon monoxide	PP3 Long-Life	gas	04	
			clear	84	
MCT-550	flood	CR2	wet	04	same transmitter as MCT-302T
			dry	84	
NEXT PIR MCW	movement	CR123A	movement	0C	
			clear	8C	
CLIP T MCW	movement	CR123A	movement	0C	3 minutes before re-triggered

These transmitters send out a 'supervisory' signal every 15 minutes or so. This is actually two signals: the first contains a code indicating the current status of the sensor; the second contains the same code but has 800000 added to the address (e.g. DB11AE instead of 5B11AE). If the tamper alarm is set off by opening the device, 40 is added to the sensor code (e.g. C4 instead of 84).

2.15 X10 Modules

2.15.1 X10 Computer Modules

The DCIU computer module is made by Domia (http://www.domia.co.uk). This is broadly equivalent to CM11U and CM12U modules from other manufacturers. This uses a Prolific PL2303USB interface chip, with drivers available from:

http://www.prolific.com.tw/eng/downloads.asp?ID=31

The computer module has been tested on Raspberry Pi, Windows XP , Windows 7 32-bit/64-bit.

When plugged in, the receiver appears as a 'Prolific USB-to-Serial Comm Port' in Windows Device Manager. Note the COM port number that is allocated to it.

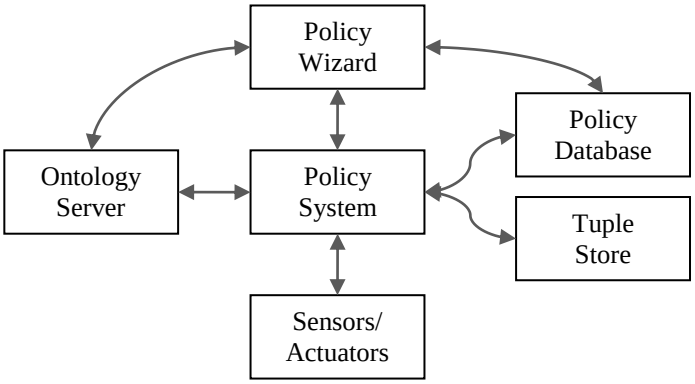
2.15.2 X10 Appliance Modules

These come from a variety of different manufacturers. Each module must be set to have a unique house address: house/room code (letters A to P) and unit/device code (numbers 1 to 16). Available modules include the following:

Model	**Function**
AD10	DIN rail switch (on, off, 16A)
AM12U	appliance module (on, off)
CM12U	computer module (RS232, USB)
LD11	DIN rail dimmer (on, off, dim, 700W)
LM12U	light module (on, off, dim)
SS13U	wall switch (3 × on, off, dim)
TM13U	transceiver module (on, off, dim?)

3 Software

A simplified architecture for the policy system is given below:



For home care, input and output are supported by the driver bundles described below.

3.1 Java Run-Time Environment

3.1.1 Microsoft Windows

A JRE is required to run Knopflerfish (32-bit or 64-bit); this could be from Oracle:
http://www.oracle.com/technetwork/java/javase/downloads

or OpenJDK:
http://openjdk.java.net

Versions 6 and 7 both work. The libraries (DLLs) loaded by drivers need to match the OS in terms of 32-bit or 64-bit version. The libraries that need to respect this are as follows:

- Audio Player: Windows 32 and 64 bit versions of Cerevoice are available
- Comm Access: Windows 32 and 64 bit versions of jspWin are available
- IRTrans Driver: Windows 32 and 64 bit versions are available
- Plugwise Driver: Windows 32 and 64 bit versions are available
- RFID Driver: Windows 32 and 64 bit versions are available for the standard *WinSCard*

On Windows, serial ports are identified as *COM1*, *COM2*, etc. including USB devices that present a serial interface. The sample property files given in this report use Windows COM port numbers and Windows paths.

3.1.2 Raspberry Pi

A JRE is required to run Knopflerfish (32-bit or 64-bit); this could be from Oracle:
http://www.oracle.com/technetwork/java/javase/downloads

or OpenJDK:
http://openjdk.java.net

Versions 6 and 7 both work, though note that only a headless version is available from Oracle for ARM v6 (i.e. no GUI classes are supported such as the Swing ones). However, the version 8 early access release works fully using the hard floating-point version of Raspbian:
http://jdk8.java.net/download.html

On Linux, serial ports are identified as */dev/ttyNN*, etc. USB devices that present a serial interface are identified as */dev/ttyUSB0*, */dev/ttyUSB1*, etc. For the Raspberry Pi, USB0, etc. should be used for serial port numbers and Unix paths should be used in property files.

3.2 Java Serial Port Support

3.2.1 Microsoft Windows

For Windows XP, download JavaComm for Windows from:

http://code.google.com/p/smslib/downloads/detail?name=javacomm20-win32.zip

Place the following files inside the current JRE. (If a JDK is in use, it is essential to place the files inside the JRE that comes with this, otherwise the driver will not be automatically loaded.)

```
bin/win32com.dll
lib/comm.jar (or lib/ext/comm.jar)
lib/javax.comm.properties
```

The JavaComm driver does not work reliably on Windows 7. It is therefore preferable to use SerialPort (http://www.serialio.com). As appropriate, a 32-bit or 64-bit version of *Lib\WinDesk\jspWin.dll* should be placed into *C:\Windows\System32*. On a 64-bit Windows system, Java may fail to load the DLL from this location; it may therefore be necessary to place it in the current user's home folder (e.g. *C:\Users\kjt*). Note that a version of *jspWin.dll* will be needed that recognises normal serial ports as well as modem ports (e.g. for a GPRS modem). The files *jspComm.jar* and *Serialio.jar* should be placed into *jre\lib\ext* (e.g. *C:\Program Files\Java\jre6\lib\ext* and *C:\usr\local\jdk\jre\lib\ext*).

In theory it is possible to use RxTx (http://rxtx.qbang.org) but the Windows driver does not yet seem to be stable. RxTx is compatible with JavaComm, but the relevant package is named *gnu.io* rather than *javax.comm*.

3.2.2 Raspberry Pi

On the Raspberry Pi, SerialPort and RxTx can again be used. RxTx has been used with an ARM v6 serial port library. The procedure for downloading and installing this is described at:
https://blogs.oracle.com/jtc/entry/serial_port_communication_for_java

Serial port support is relevant for CommAccess and for many other bundles. The code is set up for use of JavaComm/SerialPort. To use RxTx in a bundle, under its project properties for libraries choose the *rxtx* JAR in place of the *serialPort* one. Change import statements in relevant source files to use *gnu.io* in place of *javax.comm*. After compiling the code, update the bundle manifest to import and export *gnu.io* instead of *javax.comm*.

3.3 Policy Database

The policy database (not a bundle) is a standard relational database (MySQL). As this is not a bundle, it is installed and configured outside OSGi. Login and privileges need to be set up for the *home_care* user. The *home_care* database needs to be set up with the users table. The latter defines *admin* and other users. SQL scripts are provided in the 'lib' directory to automate this setup.

See [2] for more detailed information about how the policy database is used.

3.4 OSGi Bundles

3.4.1 Knopflerfish Overview

Apart from the policy database, the ACCENT components are bundles deployed in the Knopflerfish OSGi implementation (http://www.knopflerfish.org).

Knopflerfish is normally run in the foreground with a GUI, e.g. by double-clicking on *framework.jar* in the *knopflerfish/osgi* directory. However, it is also possible to run Knopflerfish in the background. A Windows service can be installed with Windows *sc* or CygWin *cygrunsrv*. The following is an example of using the latter:

```
cygrunsrv -I Knopflerfish -p C:/usr/local/jdk/bin/java.exe
  -a "-jar C:/usr/local/knopflerfish/osgi/framework.jar"
  -c C:/usr/local/knopflerfish/osgi
```

This creates a log file in *C:/var/log/Knopflerfish.log*.

All bundles are configured by means of a property file (located in, say, C:/usr/local/Knopflerfish/accent). The location of these property files is defined by an entry in *init.xargs* and *restart.xargs*, or in *fwdir/fwprops.xargs.* In addition is necessary that bundles be unpackaged from their JARs (for NabaztagDriver and OntologyServer at least). The following properties must therefore be defined in the relevant *xargs* files:

```
uk.ac.stir.cs.accent=C:/usr/local/knopflerfish/accent
org.knopflerfish.framework.bundlestorage.file.always_unpack=true
```

Despite the lack of an associated terminal, Knopflerfish manages to start up and run as a service in the background. However, in the absence of a console it is not possible to manage the framework or to see Knopflerfish logs. The Telnet Console (which runs by default) can be used with:

```
telnet localhost 2323
```

and is configured with the following typical properties (in *init.xargs* and *restart.xargs*, or in *fwdir/fwprops.xargs*):

```
org.knopflerfish.consoletelnet.port=2323
org.knopflerfish.consoletelnet.user=admin
org.knopflerfish.consoletelnet.pwd=------
```

The first three parameters allow a remote user to manage the framework and to see KF logs. The HTTP Console (not installed by default, but available as *httpconsole* from OBR 'no category') can also be installed. This is accessed in a web browser as follows, supplying the defined username and password:

```
http://localhost:8080/servlet/console
```

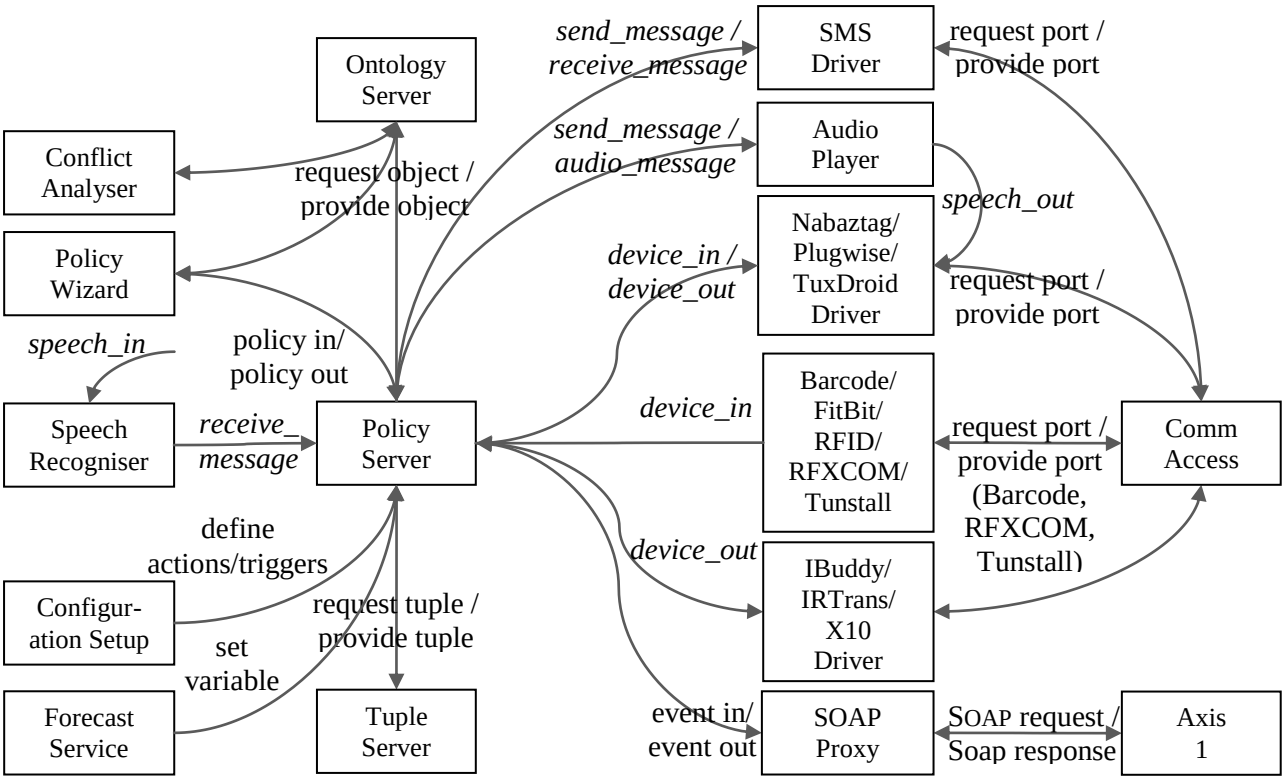
and is configured with the following typical properties in (in *init.xargs* and *restart.xargs*, or in *fwdir/fwprops.xargs*):

```
org.knopflerfish.httpconsole.requirelogin=true
org.knopflerfish.httpconsole.user=admin
org.knopflerfish.httpconsole.pwd=-----
```

This allows bundles (but not the framework or logs) to be manipulated.

3.4.2 Bundle Overview

The relationship among the ACCENT bundles is shown below. Names in italics are those of events.



The bundles communicate via events, mediated by the OSGi Event Admin service. The bundles are mostly parameterised by Java property files located in the Knopflerfish root OSGi directory (e.g. *knopflerfish/osgi*). The start levels are as follows:

Level	Bundle
6	Xerces-J
7	Axis1
7	CommAccess
7	ConfigurationSetup
7	FitBitDriver
7	IBuddyDriver
7	NabaztagDriver
7	OntologyServer
7	Pax Web Jetty
7	Pax Web JSP
7	Pax Web WAR

7	SpeechRecogniser
7	TupleServer
7	TuxDroidDriver
8	BarcodeDriver
8	IRTransDriver
8	PlugwiseDriver
8	RFIDDriver
8	RFXCOMDriver
8	SMSDriver
8	SoapProxy
8	TunstallDriver
8	X10Driver
9	PolicyServer
10	AudioPlayer
10	ConflictAnalyser
10	ForecastService
10	PolicyWizard

Italicised items in the table above are additional bundles needed for the policy wizard. Depending on the Knopflerfish start level, it may be necessary to start levels 7 to 10. Policies can be edited with the wizard as described in section 3.4.16.

3.4.3 Axis 1

The Axis bundle to support web services is adapted from code by provided by Knopflerfish. Install it in Knopflerfish with start level 7 (so Desktop is initialised first).

3.4.4 Audio Player

AudioPlayer plays pre-recorded speech created using the policy wizard and also performs TTS (Text To Speech). Install AudioPlayer in Knopflerfish with start level 10 (as it depends on PolicyServer). Audio is played by the policy action *send_message(recipient, message)*. The bundle uses the working directory *AudioPlayer* located in the ACCENT properties directory. This working directory is created automatically if needed, and is used for temporary synthesised speech files. It is possible to output pre-recorded clips and synthesised speech, either to the default audio device or to a specified one.

The policy server converts a *send_message* action into an *audio_message* event that AudioPlayer listens for. If the speech is to be synthesised into a file, AudioPlayer sends an *audio_file* event that provides the recipient (e.g. *nabaztag*) and the full path to the speech file. The intention is that another bundle for this recipient will output the speech file, which must then be deleted to avoid temporary files building up.

If the recipient is *audio* or *audio:default* then output is to the default audio output device. Any other *audio:destination* recipient causes an audio file (8-bit, 8kHz WAV file) to be created in the *AudioPlayer* directory. The name of this audio file has the format <*recipient*>< *msec*>.wav. For example, a message to *audio:nabaztag* might result in *AudioPlayer*/*nabaztag1390674487523.wav*. The creation of an audio file is notified by a *speech_out* event with *recipient* as the intended device (e.g. *nabaztag*) and *message* as the full path to the file. The receiving bundle should output this audio file and then delete it.

If *message* has the form *!text*, it is checked whether *text* names a policy variable. If so, its value is taken as a pre-recorded audio clip (16-bit, 8kHz WAV file). Otherwise *message* is treated as text to be synthesised using TTS. Such a message can start with *!voice* to indicate a voice other than the default specified in the properties file. (Such a message cannot be the name of a policy variable.) The message can contain speech markup such as '|', '||' or '|||' for a short, medium or long pause. An example message would be *'!Sarah Welcome home. || The house temperature is :interior_temperature.'*.

The property file *AudioPlayer.properties* defines the audio configuration. For TTS the bundle requires an installation of the Cerevoice Text-to-Speech SDK (http://www.cereproc.com/products/sdk). If this is not available, only pre-recorded clips can be played back. In such a case, only the *preamble.tone* property should be defined. An example property file is as follows:

```
# The owning entity for audio clip variables (user@domain):

audio.entity            admin@cs.stir.ac.uk

# The licence file for Cerevoice (default no TTS):

licence.file            C:/usr/local/cerevoicej/licence.lic
```

```
# The directory for the dynamic libraries to support Cerevoice (default no TTS):

library.directory       C:/usr/local/cerevoicej

# The directory for Cerevoice voice files (default no TTS):

voice.directory         C:/usr/local/cerevoicej

# # The default voice when not specified in the request (Heather - default,
# Jack, Sarah, Stuart):

voice.default           Stuart

# Some wireless speaker systems go to sleep when not receiving audio and can
# take a few seconds to wake up when audio is played back. If the name of a
# preamble tone is given, this is played first (current choices bell, cuckoo,
# sleigh, 20khz). If the property is not defined then there is no preamble.

preamble.tone           cuckoo
```

A request to output an audio message from the policy server can be simulated through OSGi. Choose the *Events* tab in Knopflerfish and click the *Send* button. Set something like following, then click the *Send* button.

```
Event Topic      uk/ac/stir/cs/accent/audio_message
recipient        default
message          !Sarah, welcome home. || Had a good day?
```

Speech file output from AudioPlayer can be simulated through OSGi. Choose the Events tab in Knopflerfish and click the Send button. Set something like following, then click the Send button.

```
Event Topic      uk/ac/stir/cs/accent/speech_out
recipient        nabaztag
message          C:/knopflerfish/accent/AudioPlayer/nabaztag1390674487523.wav
```

3.4.5 Barcode Driver

For a barcode reader like the Cipherlab 1070 to appear on a COM port, a driver needs to be installed. The Cipherlab installation disc and web site (*www.cipherlab.co.uk)* provide the CP210x driver from Silicon Labs. Install BarcodeDriver in Knopflerfish with start level 8 (so CommAccess is initialised first). The property file *BarcodeDriver.properties* defines the driver configuration.

The properties file allows zero or more barcodes to be mapped to the corresponding item descriptions. This is useful for locally produced barcodes (e.g. to indicate a particular programme to be recorded or an appointment to be kept). It is also useful for short-form barcodes used by supermarket own-brand goods, when the normal UPC (Universal Product Code) may not be used. If a barcode is not found in the properties file, it is looked up online. Most online barcode services are commercial. BarcodeDriver is designed to work with the free *www.updcatabase.org*. This requires registration to obtain a developer key that is used in queries to the site. If the barcode is not known to the site, BarcodeDriver will provide the literal barcode (which may still be useful in a policy). However, it is possible for anyone to define new barcodes on the site.

Barcode readings result in input events of the form *device_in(reading,barcode,instance,,description).* The instance is normally *single* for a barcode reading. However, BarcodeDriver allows the same barcode to be scanned twice (or more) in quick succession. In this case, the instance is *multiple*. This is intended for situations like bringing items into the house (a single scan) and disposing of them after use (repeated scans). This allows stocks of items to be maintained. The period within which multiple scans are recognised is defined in the properties file.

```
# The port should be the serial port where the barcode reader can be found
# (appears in Windows Device Manager under Ports as "Silicon Labs CP210x USB
# to UART Bridge"):

barcode.port     COM10

# The owning entity on whose behalf events will be triggered (user@domain):

barcode.entity   admin@cs.stir.ac.uk

# The URL used to check barcodes (default "http://www.upcdatabase.org/api/json")
```

```
barcode.url      http://www.upcdatabase.org/api/json

# The API key suffixed to the URL to identify the developer

barcode.key      developer_key

# The period within which repeated scans of a barcode are allowed (seconds,
# default 2)

barcode.period   2

# The mapping translates each barcode into a string reported when the barcode is
# read. If a barcode is not found here, it is checked with www.upcdatabase.org.
# If it is not found there, the literal barcode is reported. The format of
# entries is:
#
#   key:   barcode
#   value: item description (escape special characters, e.g. "\'")

25133707   Aldi Sweet Harvest sweetcorn
25211276   Bramwell's Real Mayonnaise
25213355   Stonemill Table Salt
25114515   Lacura Baby Lotion
20214029   Crusti Croc Cheese and Onion Crisps
25235043   Grandessa Smooth Peanut Butter
27013120   Wickes 40W R50 SES Light Bulbs
```

Input from a barcode to the policy server can be simulated through OSGi. Choose the *Events* tab in Knopflerfish and click the *Send* button. Set something like following, then click the *Send* button.

```
Event Topic      uk/ac/stir/cs/accent/device_in
user             admin@house.stir.net
arg1             reading (message type)
arg2             barcode (entity name)
arg3             single or multiple (entity instance)
arg5             description or numerical identifier (parameter values)
```

3.4.6 Comm Access

The communications port access bundle is adapted from code by Michael Wilson. Install it in Knopflerfish with start level 7 (so Desktop is initialised first). It opens the serial ports defined by the property file *CommAccess.properties*, e.g.:

```
# Comma-separated list of ports to offer via services (the actual ports
# may be a subset of this list). Spaces around commas are ignored:

comm.ports COM9,COM10,COM11,COM12
```

The actual serial ports opened from this list will depend on what is available. CommAccess provides access to ports via a service. It may be necessary to restart this bundle if serial ports become unavailable. For example, another bundle that uses a serial port from CommAccess may (incorrectly) close the port.

3.4.7 Configuration Setup

The configuration setup bundle is installed in Knopflerfish with start level 7 (so Desktop is initialised first). It reads the property files of all relevant bundles (IRTransDriver, PlugwiseDriver, RFXCOMDriver, X10Driver). It then updates the PolicyWizard property file with lists of corresponding device actions and triggers. After performing this, the bundle stops. It could therefore be set to run on every startup on Knopflerfish.

The bundle is configured by the property file *ConfigurationSetup.properties*:

```
# Policy domain to be configured

policy.domain          home_care
```

This utility is normally run as a bundle, but can also be run as an application. In the latter case, certain constants in the code may need to be adjusted (the location of property files and the policy domain).

3.4.8 Conflict Analyser

The offline conflict analyser is developed from code by Gavin Campbell. It will also run as a Java application. Install it in Knopflerfish with start level 11 (so the Ontology Server and Policy Server are initialised first). The conflict analyser is configured by the file *ConflictAnalyser.properties*, e.g.:

```
# Name of the policy server host (e.g. "localhost")

policy.host             localhost

# Port used for for uploading policies (e.g. 9999)

policy.port             9999

# Resolution policy owner

policy.owner            admin@cs.stir.ac.uk

# Name of the ontology server host (e.g. "localhost")

poppet.host             localhost

# Whether to overwrite an existing resolution ("true" or default "false")

overwrite.resolution    true
```

3.4.9 FitBit Driver

The FitBit driver should be installed in Knopflerfish with start level 7. The driver is configured by the file *FitBitDriver.properties*, e.g.:

```
# The owning entity on whose behalf events will be triggered (user@domain):

fitbit.entity           admin@cs.stir.ac.uk

# Interval between FitBit data retrieval (hours, default 3)

fitbit.interval         3

# Access token allocated by FitBit for retrieving data. If this value is not
# given, the bundle will output instructions on how to get an access token.

access.token            eyJAaGciOiJIU...Are0t1zQo9bG

# Client identifier allocated by FitBit for this application (default
# "client.id", used only when instructions are needed about how to get an
# access token)

client.id               381AB4
```

3.4.10 Forecast Service

This bundle uses external forecasting services for air pollution(DEFRA, http://uk-air.defra.gov.uk), pollen(MeteoVista, http://www.meteovista.co.uk), and weather (OpenWeatherMap http://www.openweathermap.org). However this means that, unlike most other bundles, the forecast bundle requires the ACCENT system to have an Internet connection. It retrieves forecasts and store them as text in the *exterior_pollution*, *exterior_pollen* and *weather_forecast* policy system variables. The value of these variables can then be used for, say, advising the user to close windows or to speak a forecast. Install the bundle in Knopflerfish with start level 10 (as it relies on PolicyServer running to set the variable). The bundle is configured by the property file *ForecastService.properties*, e.g.:

```
# The owning entity on whose behalf forecasts are stored (user@domain):

forecast.entity  admin@cs.stir.ac.uk

# Interval between forecast checks (hours, default 3)
```

```
forecast.interval       3

# Pollen forecast location (according to http://www.meteovista.co.uk, e.g.
# "Stirling/4409327")

pollen.location         Stirling/4409327

# Pollution forecast location (according to http://uk-air.defra.gov.uk, e.g.
# "56.12,-3.94" for Stirling, meaning 56.12 degrees North and 3.94 degrees West)

pollution.location      56.12,-3.94

# Weather forecast location (according to http://openweathermap.org,
# "<city>,<country>" e.g. "Stirling,UK", or location identifier e.g. "2636910")

weather.location  Stirling,UK

# Weather forecast user identifier (if required by the weather service)

weather.id              754d06555
```

3.4.11 i-Buddy Driver

The i-Buddy bundle should be installed in Knopflerfish at start level 7. It is configured by the property file *IBuddyDriver.properties*, e.g.:

```
# In the following, alternatives are separated by '|'. Internal action
# parameters are the same as policy parameters.

# Reinitialise everything on the i-Buddy

reset                                                   reset

# Flash the i-Buddy head a pair of colours for a fixed time

flash,head,blue+white|cyan+white|green+white|purple+white|red+white|yellow+white
  flash,head

# Set the i-Buddy heart off, on or to flash

flash|off|on,heart                                      flash|off|on,heart

# Set the i-Buddy head off or on for a fixed time

off|on,head,blue|cyan|green|purple|red|yellow|white     off|on,head

# Set or flap the wings

set,wings,down|up|flap_fast|flap_slow                   set,wings

# Spin operations that can be performed

rotate,body,left|right|spin                             rotate,body
```

3.4.12 IRTrans Driver

The infrared transceiver bundle should be installed in Knopflerfish with start level 8 (so CommAccess is initialised first). Note that an IRTrans server instance must already be running on the target host, and that this must have definitions compatible with the defined protocol commands. It uses a remote IRTrans server defined by the property file *IRTransDriver.properties*, e.g.:

```
# Server host name and port number for the system running the IRTrans server:

ir.server.name          localhost
ir.server.port          21000

# Port number for the system running the IRTrans client:
```

```
ir.client.port          21001

# The following mapping describes a comma-separated list of mappings from
# policy actions to protocol commands. The key data must match policies in
# respect of case, but the value data can be in either case. Spaces can be used
# after commands, and special characters such as "=" must be escaped as "\=".
#
#   key:   message,entity,instance,parameters
#   value: command,address,parameters
#
# One or more messages or commands may be given, separated by "|"; the number of
# these in the key and value must be the same. The instance is optional. The
# parameters can be comma-separated and are optional; if a policy action does
# not match the mapping with its specific parameters, a match is tried without
# the parameters.
#
# Parameters then instance may be omitted from the right (e.g.
# "message,entity,instance" and "message,entity" or "command,address" can be
# used).
#
# Mapping entries can be repeated for the same message, entity and instance but
# with different parameters.

# All devices support "off", "on" and "on!" (power-up) commands. Specialised
# commands are as follows:
#
#   CD player:          back (slow), forward (slow), next (track), pause, play,
#                       previous (track), stop, track_set (1 or 2 digit track
#                       number parameter)
#
#   DVD/VHS Recorder:   back (slow), channel_down, channel_set (1 or 2 digit
#                       channel number parameter), channel_up, drive (DVD, VHS),
#                       forward (slow), mute, next (track), pause, play,
#                       previous (track), record, stop, volume_down, volume_up
#
#   TV, FreeView: mute, channel_down, channel_set (1 or 2 digit channel
#                       number parameter), channel_up, volume_down, volume_up
#

# TV in lounge
#  o channels 10 and above need two digits in quick succession

off|on,tv,lounge                                    off|on!,tv_lounge

volume_down|volume_up|mute,tv,lounge                vol-|vol+|mute,tv_lounge

channel_down|channel_up|channel_set,tv,lounge       chan-|chan+|chan!,tv_lounge

# CD player in lounge
#   o tracks 10 and above need NN and then two digits in quick succession
#   o back/prev need to be repeated to keep searching back/forwards

off|on,cd,lounge                                    off|on,cd_lounge

back|forward|previous|next,cd,lounge                back|fwd|prev|next,cd_lounge
play|pause|stop,cd,lounge                           play|pause|stop,cd_lounge

track_set,cd,lounge                                 track!,cd_lounge

# DVD recorder in lounge
#   o channels 10 and above need two digits in quick succession

off|on,dvd,lounge                                   off|on,dvd_lounge
```

```
back|forward|previous|next,dvd,lounge              back|fwd|prev|next,dvd_lounge
drive,dvd,lounge                                   drive,dvd_lounge
play|record|pause|stop,dvd,lounge                  play|rec|pause|stop,dvd_lounge

channel_up|channel_down|channel_set,dvd,lounge     chan+|chan-|chan!,dvd_lounge
```

Output to an IR device from the policy server can be simulated through OSGi. Choose the *Events* tab in Knopflerfish and click the *Send* button. Set something like following, then click the *Send* button.

```
topic           uk/ac/stir/cs/accent/device_out
arg1            pause (message type)
arg2            CD (entity name)
arg3            lounge (entity instance)
```

3.4.13 Nabaztag Driver

This bundle bidirectionally interacts with an Nabaztag 'Internet rabbit'. Besides being a bundle, it can also be run as a Java application (using *jNabServer.jar*). The bundle interacts with a modified version of jNabServer for managing the Nabaztag (http://code.google.com/p/jnabserver). This effectively replaces the original Violet server with something roughly equivalent.

The system that runs the server dictates the IP address to be used by the Nabaztag. jNabServer can use a variety of ports, but it is best to avoid common ones such as 80 or 8080. The choice of port may also be affected by the local firewall.

To set up the Nabaztag, press and hold the head button while powering the rabbit on – the lights will turn blue. This creates a wireless network called *nabaztagNN*. Connect to this, and go to 192.168.0.1 in a web browser. Configure the network parameters, ideally using DHCP from the wireless router. The rabbit should be set to connect to the system that will run the server. For example, this might use *192.168.0.15:8181/vl* (the *vl* for manufacturer Violet being needed). The Nabaztag appears to work with WPA and WPA2 using PSK or TKIP+AES, though other combinations may work. This may require the wireless router security to be set appropriately. Once the rabbit has been configured, it will connect to the selected server system. This runs a servlet web container that responds to requests for JSPs.

Rabbits are identified by serial number, which is the same as their wireless MAC address. jNabServer supports a configuration server that allows rabbits to be given a friendly name. Typically connect to the configuration server using *telnet* to port 6969.

For use with ACCENT, a Nabaztag is automatically associated with *AccentPlugin* when it first contacts the server. This plugin interacts bidirectionally with the bundle, allowing policy triggers to be provided and policy actions to be performed.

The bundle uses the working directory *NabaztagDriver* located in the ACCENT properties directory. This working directory is created automatically if needed. The directory is used to hold plugins (as defined by the bundle build – currently only *AccentPlugin*) and Nabaztag boot code (both copied from the bundle contents). It is also used for temporary audio input files that are created on audio input. The *save* command of the configuration server preserves the state of a Nabaztag in *bunnies/<serialno>.ser*. A choreographies sub-directory is created but is currently not used by the bundle.

The bundle is defined by the property file *NabaztagDriver.properties*, e.g.:

```
# The owning entity on whose behalf events will be triggered (user@domain):

nabaztag.entity                     admin@cs.stir.ac.uk

# Port number for the jNab configuration server

nabaztag.configuration.port         6969

# Port number for the jNab rabbit server

nabaztag.rabbit.port                8181

# Debug setting ("debug", "error", "info" (default))

nabaztag.log.level                  info

### Input triggers ###

# Button click on head
```

```
click,head                              click,head
double_click,head                       double_click,head

# Ear position

moved,left_ear,back|down|forward|up   moved,left_ear
moved,right_ear,back|down|forward|up  moved,right_ear

# RFID tag (reported as a "reading" event)

d0021a053b4702e7                        Weather
d0021a053b45307b                        Medicine
d0021a053b452e48                        Cooking

### Output actions ###

# Lights

flash|off|on,belly,blue|green|orange|purple|red|white|yellow
  flash|off|on,belly
flash|off|on,bottom,blue|green|orange|purple|red|white|yellow
  flash|off|on,bottom
flash|off|on,center,blue|green|orange|purple|red|white|yellow
  flash|off|on,center
flash|off|on,centre,blue|green|orange|purple|red|white|yellow
  flash|off|on,centre
flash|off|on,left,blue|green|orange|purple|red|white|yellow
  flash off|on,left
flash|off|on,nose,blue|green|orange|purple|red|white|yellow
  flash|off|on,nose
flash|off|on,right,blue|green|orange|purple|red|white|yellow
  flash|off|on,right

# Ears

move,left_ear,back|down|forward|out|up            move,left_ear
move,right_ear,back|down|forward|out|up           move,right_ear
move,both_ears,back|down|forward|out|up           move,both_ears
```

Speech output to the Nabaztag driver can be simulated through OSGi. Choose the *Events* tab in Knopflerfish and click the *Send* button. Set something like following, then click the *Send* button.

```
Event Topic      uk/ac/stir/cs/accent/speech_out
recipient        nabaztag
mesage           C:/knopflerfish/accent/002185ba6790.wav
```

3.4.14 Ontology Server

The ontology server bundle (POPPET) is installed in Knopflerfish with start level 7 (before the policy server and policy wizard are started). It will also run as a Java application. The ontology server starts an RMI Registry instance on the same system. Install the ontology server in Knopflerfish with start level 7 (before PolicyServer is initialised).

In principle the ontology server can be run on a remote system. In practice, a firewall and RMI security can get in the way. At the least, the firewall of the remote system must allow incoming connections on port 1099. A security manager and security policy could also be need on the remote ontology server, though even these can lead to access issues. For these reasons, it is recommended to run the ontology server on the local host.

The ontologies themselves can reside on a remote system or on the local host. For example, they might be deployed in *Tomcat/webapps/schemas*. In this case, the properties file and the imports in the OWL files should refer to *localhost:8080/schemas*.

Note that the Ontology Server needs to be built as an OSGi bundle (for use in Knopflerfish), and also as a JAR file (for use from the command line) using the Ant build file. If the Poppet JAR file is updated, it needs to be replace the version currently in the Policy Wizard *WEB-INF/lib* directory. The Unix scripts *poppet-start* and *poppet-stop* are provided in *accent/bin* for command line use. See [1] for more detailed information about the ontology server.

The ontology server supports an ontology service defined by the property file *OntologyServer.properties*, e.g.:

```
# Base URL for OWL ontologies ("http://" prefix assumed)

ontology.base                  www.cs.stir.ac.uk/schemas

# Comma-separated list of domains to support (currently call_control, home_care
# and sensor_network). Spaces around commas are ignored:

ontology.domains               home_care.
```

3.4.15 Policy Server

The policy server can run as a bundle or as a Java application. Install it in Knopflerfish with start level 10 (so OntologyServer, PolicyAction and TupleServer are initialised first). If OntologyServer or TupleServer is restarted while the policy server is running, this can cause bizarre effects (e.g. the policy server upload port is not properly closed).

The bundle normally uses the working directory *PolicyServer* located in the ACCENT properties directory. This working directory is created automatically if needed and is typically used for a log file.

The policy server is configured by file *PolicyServer.properties*. The standard settings are for trigger/action communication, policy upload/download, email, database access, tuple space access, and event mechanism.

```
# Policy Server properties when running on the local machine

# Database server name (e.g. "localhost"), remote access port number (e.g. 3306)

database.server       localhost
database.port         3306

# Accent database name (e.g. "accent") , username for accessing this
# (e.g. "accent"), password for accessing this, terminology mapping table name
# (e.g. "terminology_mapping")

database.name         accent
database.username     accent
database.password     ------
database.table        terminology_mapping

# Policy event handler (e.g. "EventAdmin" for plain OSGi, "MessageBroker" for
# MATCH" or "ContextServer")

event.provider        EventAdmin
# event.provider      MessageBroker
# event.provider      ContextServer

# Application domain and administrator email for this

application.domain    home_care
application.owner     admin@cs.stir.ac.uk

# Email server (e.g. "smarthost.cs.stir.ac.uk"), SMTP port number (e.g. "25"),
# email user address and password, subject used for email messages from the
# system

mail.server           smarthost.cs.stir.ac.uk
mail.port             25
mail.user             kjt@cs.stir.ac.uk
mail.password         ------
mail.subject          Policy System Message

# Prefix of system policies (instantiated prototypes) and system variables
# (e.g. "!"), prefix of prototype parameters (e.g. "$"), prefix for variables
# (e.g. ":", which must be escaped as "\:")

parameter.prefix      $
```

```
system.prefix         !
variable.prefix       \:

# Port used for sending events to the policy server (e.g. 9998) and port used
# for uploading/querying policies (e.g. 9999)

policy.message.port   9998
policy.upload.port    9999

# Directory (relative to policy server root) for logging

policy.log.directory  PolicyServer

# Name of the ontology server host (e.g. "localhost")

poppet.host           localhost

# Hex flags to turn on debugging (0000 typically, 0400 to allow wizard policy
# checking, 0404 to add triggered policies, 06CD to add goal handling,
# 07FF for everything):
#   0000  report nothing
#   0001  report prototype contributions to goals
#   0002  report triggering status for policies
#   0004  report triggered policies
#   0008  report dynamic analysis summary
#   0010  report goal formula
#   0020  report identical scores for different prototype combinations
#   0040  report prototype indexes and names being considered for goals
#   0080  report optimised policies
#   0100  report defuzzified policies
#   0200  report static analysis summary
#   0400  record activations even if simulated (for wizard Check Policies)

server.debug            06CD

# Number of lines to preserve in a server log (default 2048)

server.lines          2048

# Number of activation/history records to preserve in the server journal
# (default 128)

server.records          128

# Name of the tuple server host (e.g. "localhost"), port number (e.g. "8200")

tuples.server         localhost
tuples.port           8200

# Name of the policy database (e.g. "Policies"), policy username (e.g.
# "accent"), and password

tuples.name           Policies
tuples.username       accent
tuples.password       ------
```

See [2] for more detailed information about the policy server.

3.4.16 Policy Wizard

The policy wizard bundle allows easy definition and editing of policies, goals, etc. It is possible to run the policy wizard as a Tomcat web application. This requires a Tomcat context file *Tomcat/conf/Catalina/localhost/wizard.xml* such as:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context docBase="C:/Users/kjt/Home/bin/accent/PolicyWizard"
```

```
   path="/wizard" reloadable="true"/>
```

However, if Tomcat is in use then Pax Web must not be simultaneously running in Knopflerfish as the same port is used by both. For home care, the policy wizard is typically accessed as follows (the trailing '/' is optional with Tomcat but required for Pax Web):

```
http://localhost:8080/wizard/home_care/
```

OntologyServer, PolicyServer and TupleServer must be started before the policy wizard is started. (If any of these is restarted with the policy wizard is running, it will be necessary to restart from the policy wizard login page.) Install it in Knopflerfish with start level 11 (so that PolicyServer is initialised first).

Normally the policy wizard would be run in Knopflerfish using Pax Web. The HTTP-Server and HTTP-Root-Impl bundles need to be stopped before running Pax Web as the latter will install Jetty as a replacement servlet container. The policy wizard runs as a set of JSPs using Pax Web (http://wiki.ops4j.org/display/paxweb/Pax+Web). Specifically, the policy wizard needs the Pax Web bundles for Jetty, JSP and WAR Extender. It also needs the Xerces-J bundle from Knopflerfish.

Note that the Policy Wizard needs to be built as an OSGi WAR bundle (for use in Knopflerfish) and also as a webapp (for use in Tomcat) using the *build* script in the top-level Policy Wizard directory. See [4] for more detailed information about the policy wizard.

The policy wizard always has an *admin* user who can create other users. The email address for one of these must be the same as that provided to various communications drivers (e.g. admin@house.stir.net).

When policies are formulated, their triggers and actions must match the mappings in various communications drivers and PolicyAction. Sample triggers are as follows:

```
when told of unoccupied by bed

when told of movement in lounge
```

Sample actions are as follows:

```
do perform dim of light in bathroom with value 20

do perform on of heating in kitchen

do perform pause of CD in lounge
```

Although the policy wizard normally runs as a bundle, it is configured outside OSGi. A WAR file, with OSGi bundle manifest, is created by running the following command in the policy wizard root directory (e.g. after recompiling the code or changing property files):

```
jar cfm bin/PolicyWizard-2.0.0.war META-INF/MANIFEST.MF call_control home_care
  sensor_network WEB-INF
```

The location of the property files depends on whether the wizard is running as a bundle (system property *uk.ac.stir.cs.accent* is defined) or as a normal webapp (this property is not defined). In the following, *root* might be *C:/usr/local/knopflerfish/accent*, *domain* might be *home_care*, and *language* might be *en-GB*.

- Bundle: The database property file is external to the wizard, the mapping and wizard property files are bundled with the wizard and are therefore internal:
  *root/*PolicyWizard.*domain*.database.properties
  PolicyWizard/WEB-INF/lib/*domain*.mapping.properties
  PolicyWizard/WEB-INF/lib/*domain/language/*wizard.properties
- Webapp: The database, mapping and wizard property files are bundled with the wizard and are therefore internal:
  PolicyWizard/WEB-INF/lib/*domain/*database.properties
  PolicyWizard/WEB-INF/lib/*domain/*mapping.properties
  PolicyWizard/WEB-INF/lib/*domain/language/*wizard.properties

The file *database.properties* defines interfaces to other servers (policy database, policy server, ontology server), e.g.:

```
# Home care wizard properties when running on local machine

# System administrator email address

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

# Name of the database host (e.g. "localhost"), username (e.g. "home_care") and
# password, name of users database table (e.g. "home_care")

users.host                     localhost
```

```
users.password                ------
users.table                   home_care
users.username                home_care

# Name of the policy server host (e.g. "localhost") and upload port number
# (e.g. "9999")

policy.host                   localhost
policy.message.port           9998
policy.upload.port            9999

# Name of the ontology server host (e.g. "localhost") and ontology name
# (e.g. "home_care")

poppet.host                   localhost
poppet.ontology.name          home_care

# URL for generic, wizard and home care ontologies (note: append `#' to URI)

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/home_care.owl#

# Prefix of system policies (instantiated prototypes) and system variables

system.prefix                 !

# Hex flags to turn on debugging (0000 typically, 0001 for everything):
# 0000 (nothing), 0001 (show policy tree), 0002 (show system variables -
# '!' prefix and ontology variables - '*' prefix)

wizard.debug                  0000

# Device actions and triggers

device.actions                back,cd,lounge/.../volume_up,tv,lounge

device.triggers               active,flush,toilet/.../shut,window,lounge
```

The device action and trigger properties are currently used only for the home care domain. These are triples separated by '/'. The triples are comma-separated and have the form *message_type,entity_name,entity_instance*. Although these properties can be manually defined, they are normally created automatically by the ConfigurationSetup bundle (see section 3.4.7).

The file *mapping.properties* defines the mapping from policy language names to policy wizard names, e.g. (in part):

```
# Mapping from policy language names to natural language names

# triggers

device_in                     policy.device.in
device_in_raw                 policy.device.in.raw

#conditions

entity_name                   policy.entity.name
...

# actions

device_out                    policy.device.out
device_out_raw                policy.device.out.raw

# operators
```

```
and                             operator.and
...

ge_epoch                        operator.ge.epoch
...

# Generic conditions

date                            policy.date
day                             policy.day
time                            policy.time

# preferences

must                            policy.must
...
nothing                         policy.nothing
availablility                   policy.status.availability
presence                        policy.status.presence
availablility_raw               policy.status.availability.raw
presence_raw                    policy.status.presence.raw

# Generic policy triggers

timer_expiry                    policy.timer.expiry
timer_expiry_raw                policy.timer.expiry.raw

# Generic policy actions

log_event                       policy.log.event
...
log_event_raw                   policy.log.event.raw
...
apply_default                   resolution.apply.default.action
...
preference0                     resolution.preference0.condition
variable0                       resolution.variable0.condition
...
locale.de-DE                    language.de.de
...
stage.0                         policy.novice
...
```

The file *wizard.properties* defines the mapping from policy wizard names to natural language, e.g. (in part):

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

# Generic Interface properties

aspect.applicability            Applicability (label, owner, ...)
...
button.cancel                   Cancel
...
edit.action                     Edit Action
...
error.database                  Cannot read database
...
language.de.de                  German - Germany
...

# Generic Policy User levels/stages

policy.administrator            administrator
...
```

```
# Generic Policy Preferences

policy.must                    must
...

# Generic Wizard Hint Text

hint.action                    Set empty to remove an action
...

# Domain-Specific Wizard Hint Text
#
# e.g. for hints of a category of trigger/condition/action:
# hint.*category_name*.action.category
# hint.*category_name*.condition.category
# hint.*category_name*.trigger.category

hint.configure.action.category e.g. 'perform off at bathroom light' or
                               'perform record at VCR 2 delayed 1:00 with value
                               channel=3,period=0:30'
...

# 'status' variables (profile is a permanent variable, but hint text can be
# specified for the domain)

hint.status.profile            e.g. 'weekend' or 'emergency' (empty implies all
                               policies)
...

# Generic Operators

operator.and                   and
...

# Domain-Specific Operators

operator.ge.epoch              is or is after
...

# Generic Policy Properties

policy.address                 address
...

# Domain-Specific Policy Properties

# Triggers

policy.device.in               told of
policy.device.in.raw           device_in(type,ent,inst,period,pars)
policy.status.availability.raw availability
policy.status.presence.raw     presence
...

# Conditions

policy.entity.instance         entity instance
...

# Actions

policy.device.out              perform
policy.device.out.raw          device_out(type,ent,inst,period,pars)
```

```
# Miscellaneous, e.g. descriptive elements such units of time, quantity, etc.

policy.Kbps                    Kbps
...

# Status Variables

policy.status.availability    availability
policy.status.presence        presence

# Domain-Specific Trigger Categories and Argument Labels
#
# e.g. policy.*category_name*.trigger.category     *value*
# e.g. policy.*category_name*.trigger.arg*number*  *value*

policy.configure.trigger.category    event
...

# Domain-Specific Condition Categories
#
# e.g. policy.*category_name*.condition.category   *value*

policy.epoch.condition.category      time
...

# Domain-Specific Action Categories and parameter arg labels
#
# e.g. policy.*category_name*.action.category      *value*
# e.g. policy.*category_name*.action.arg*number*   *value*

policy.configure.action.category     perform
...
policy.send.action.category          send
...
policy.timer.trigger.arg1            called
...
policy.update.action.category  update

# Resolution Conditions (generic)

resolution.POP.condition.category    parameter/parameter comparison
...
resolution.preference0.condition     preference0
resolution.variable0.condition       variable0
...

# Resolution Actions (generic)

resolution.apply.default.action      apply default resolution
...
resolution.generic_res_action.category      generic resolution

# Resolution Action (domain-specific)

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

# Generic properties

error.action.arg.empty        Parameter field(s) for the selected action cannot
                              be empty and must not begin with "?"
...
```

```
# Domain-specific properties

error.address                 Define address in "person@domain", "sms:number" or
                              ":variable" format
...
```

3.4.17 Plugwise Driver

This bundle was written using public information about the Plugwise protocol. This information is incomplete and conjectural as no official specification exists of the protocol. Although the bundle works satisfactorily, there are some uncertainties as to its operation. Install it in Knopflerfish with start level 8 (so CommAccess is initialised first). It uses the serial port and configuration parameters defined by the property file *PlugwiseDriver.properties*, e.g.:

```
# The port should be the serial port where the Plugwise receiver can be found
# (appears in Windows Device Manager under Ports as "USB Serial Port"):

plugwise.port                 COM14

# The owning entity on whose behalf events will be triggered (user@domain):

plugwise.entity               admin@cs.stir.ac.uk

# The number of attempts allowed at sending a message (increase for a large or
# noisy network):

plugwise.retry.limit          3

# The first part of all Plugwise module addresses:

plugwise.address.start        000D6F0000

# The interval (minutes) between polls for recent energy consumption. This could
# be as frequently as required since communication errors may result in energy
# readings being lost. The result, however, would be repeated triggers for the
# same energy reading. The normal setting would, however, be for 60 minutes:

plugwise.energy.interval      60

# The interval (minutes) between polls for instantaneous power consumption:

plugwise.power.interval       10

# The mapping relates Circle addresses to entity name/instance in a policy
# trigger or action. The Circle address can be in upper or lower case, but the
# entity data must match policies in respect of case. One Circle address must be
# followed by '+' to indicate it is the Circle+. Spaces around commas in the
# entity data are ignored.
#
#   module:      Circle address (followed by '+' for Circle+)
#   policy:      entity name, entity instance

# 72AE95               stick

769F17                 kettle,kitchen
76A75A                 charger,lounge
76B60E                 lamp,lounge
76B960+                iron,lounge
76AEB5                 phone,hallway
```

3.4.18 RFID Driver

The ACS ACR122U installation disc and web site (*www.acs.com.hk*) provide drivers. Barcode readings result in input events of the form *device_in(reading,rfid,tag,,description)*. Install RFIDDriver in Knopflerfish with start

level 8 (for commonality with similar drivers, though it does not use CommAccess). The property file *RFIDDriver.properties* defines the driver configuration, e.g.:

```
# The owning entity on whose behalf events will be triggered (user@domain):

rfid.entity      admin@cs.stir.ac.uk

# The mapping translates each RFID tag UID (Unique Identifier) into a string
# reported when the tag/card is read. If the identifier is not found here, the
# literal value is reported. The format of entries is:
#
#   key:   tag/card UID as hex with upper-case letters
#   value: item description (escape special characters, e.g. "\'")

4EB41A17   Record \'East Enders\' today
8E6DE91F   Wake me in one hour
9E0C1D17   Ask my daughter to call
CEB2E81F   Cancel milk for one week
DED2E81F   Tell my doctor I am ill
```

The properties file allows zero or more card/tag UIDs (Unique Identifiers) to be mapped to the corresponding descriptions. This can be used by associating particular cards/tags with particular functions. If an identifier is not found in the properties file, it is provided in literal hexadecimal form (which may still be useful in a policy).

Input from an RFID card/tag to the policy server can be simulated through OSGi. Choose the *Events* tab in Knopflerfish and click the *Send* button. Set something like following, then click the *Send* button.

```
Event Topic      uk/ac/stir/cs/accent/device_in
user             admin@house.stir.net
arg1             reading (message type)
arg2             rfid (entity name)
arg3             tag (entity instance)
arg5             description or 4-byte hexadecimal identifier (parameter values)
```

3.4.19 RFXCOM Driver

The radio receiver bundle should be installed in Knopflerfish with start level 8 (so CommAccess is initialised first). It uses the serial port defined by the property file *RFXCOMDriver.properties*, e.g.:

```
# The port should be the serial port where the RFCOM receiver can be found
# (appears in Windows Device Manager under Ports as "USB Serial Port"):

rfxcom.port      COM12

# The owning entity on whose behalf events will be triggered (user@domain):

rfxcom.entity    admin@house.stir.net

# The mapping translates each sensor signal to trigger parameters (which can be
# omitted from the right). The sensor data can be in upper or lower case, but
# the trigger data must match policies in respect of case. In the key, no spaces
# are allowed and "=" must be escaped as "\=".
#
#   key:   sensor id,sensor code
#   value: message type,entity name,entity instance,parameter values

# If the signal from a sensor is repeated with the same parameters, it is
# ignored and no event is generated.

# Oregon Scientific temperature-humidity sensors are supported (type BTHR918,
# THGR228, THGR918). The sensor id is the device type (0X5A6D, 0X1A2D, 0X1A3D
# respectively). The sensor code is the channel number in the top quartet
# (0 = 0X0, 1 = 0X1, 2 = 0X2, 3 = 0X4), and the device function in the bottom
# quartet (0X0 = temperature, 0X1 = humidity). For example, "1A2D,01" is a
# THGR228 humidity reading. The sensor reading (temperature, humidity) is
# automatically set in the parameter values ("arg5"). Barometric pressure from
# a BTHR918 is ignored, as are signals from other types of sensors.
```

```
# All Visonic sensors are supported. The sensor id is a unique address that must
# be learned separately. For devices that send two signals, sensor code
# 0X04 = active (e.g. open, smoke/gas, wet) and 0X84 = inactive (e.g. closed,
# clear, dry). For devices that send one signal, sensor code 0X0C = active
# (e.g. movement, alarm). No parameter values ("arg5") are set.

# main bed occupancy:

4891AE,04  unoccupied,bed,main
4891AE,84  occupied,bed,main

# movement sensor:

055DCB,0C  movement,lounge

# indoor/outdoor temperature-humidity sensor (type THGR228):

1A2D,10          reading,outdoor,temperature
1A2D,11          reading,outdoor,humidity
```

Input from a wireless device to the policy server can be simulated through OSGi. Choose the *Events* tab in Knopflerfish and click the *Send* button. Set something like following, then click the *Send* button.

```
Event Topic      uk/ac/stir/cs/accent/device_in
user             admin@house.stir.net
arg1             unoccupied (message type)
arg2             bed (entity name)
arg3             main (entity instance)
```

3.4.20 SMS Driver

The SMS bundle should be installed in Knopflerfish with start level 8 (so CommAccess is initialised first). It uses a GPRS modem defined by the property file *SMSDriver.properties*, e.g.:

```
# The port should be the serial port where the GPRS modem can be found
# (appears in Windows Device Manager under Modems as "GPRS609 USB Modem";
# to find the port number, go to "Network Connections", "GPRS609
# USB", "Properties")

# The port should be the serial port where the GPRS modem can be found:

sms.port            COM9

# The bit rate for the serial port (e.g. 115200, 57600, 38400, 19200, 9600):

sms.rate            115200

# The owning entity on whose behalf events for incoming messages will be
# triggered (user@domain):

sms.entity          admin@house.stir.net

# The international prefix for the locale (e.g. 44 for UK):

sms.international   44

# The national prefix for the locale (e.g. 0 for UK):

sms.national        0

# The following mapping translates policy user addresses into mobile phone
# numbers; the reverse mapping is automatically defined. The corresponding user is
# then used as the caller in the policy trigger.
#
# If the calling phone number does not match any entry, it will be provided
```

```
# literally as the caller.
#
# A user address has the form 'user@domain'; case is significant, and spaces are
# not allowed. A mobile phone number is a list of digits, but with the following
# variants:
#
#  ' '       white space is allowed but removed
#  '-'       a minus is allowed but removed
#
# If the inbound or outbound phone number starts with "+", it will be
# converted from international to national form prior to lookup in this table.

admin@house.stir.net       07593-246-801

kjt@cs.stir.ac.uk          07811-123-456
```

When an inbound message arrives, it results in a *receive_message* trigger for the policy server with the following environment parameters:

```
user             as defined by sms.entity
call_type        "SMS"
topic            "Text message"
caller           policy user address (originating phone number if not defined)
call_content     the text message itself
```

An outbound message is sent with the action a *send_message(recipient,message)* from the policy server. The recipient can be specified as a national phone number (e.g. 07811-123-456), an international one (e.g. +44-7811-123-456), or as a user address (e.g. kjt@cs.stir.ac.uk). The latter will be converted into a national phone number using the property definitions.

A request to output an SMS message from the policy server can be simulated through OSGi. Choose the *Events* tab in Knopflerfish and click the *Send* button. Set something like following, then click the *Send* button.

```
Event Topic      uk/ac/stir/cs/accent/send_message
recipient        07811-123-456
message          Time for lunch
```

'+', '-' and white space are removed from the phone number as given. An international phone number such as '+44-7811-123-456' may also be used.

3.4.21 Soap Proxy

The SOAP Proxy maps between OSGi events and web service calls. Install it in Knopflerfish with start level 8 (so that Axis1 is initialised first). It maps events as defined by the property file *SoapProxy.properties*, e.g.:

```
# The base URL of the BPEL services, "/services/<service_name>" being appended
# to this:

proxy.url              http://localhost:8080/active-bpel

# A comma-separated list of entity names in any case that may appear in
# device_in/out events. If class <Entity>In and/or <Entity>Out exist, it is
# instantiated. The following entity types are currently recognised:
#
#   alert (in)         fall, freezing
#   bed (in)           free, occupied
#   cd (out)           back, fast_forward, fast_reverse, forward, next, off,
#                      on, pause, play, previous, record*, stop,
#                      track_set(digits)
#   chair (in)         free, occupied
#   cooker (out)       off, on
#   door (in)          open, shut
#   door (out)         lock, unlock
#   drier (out)        off, on
#   dvd (out)          back, channel_down, channel_set(digits),
#                      channel_up, drive*, fast_forward, fast_reverse, forward,
#                      mute*, next, off, on, pause, play, previous, record*,
#                      stop, volume_down, volume_up
#   dvb (out)          channel_down, channel_set(digits), channel_up,
```

```
#                       mute, off, on, volume_down, volume_up
#                       volume_up
#   fall (in)           active
#   fan (out)           off, on
#   flood (in)          clear, active
#   gas (in)            clear, active
#   heating (out)       off, on
#   humidity (in)       reading(percentage)
#   light (out)         dim(percentage), off, on
#   mat (in)            active
#   medicine (in)       active
#   message (in, out)   receive(sender,subject,message),
#                       send(sender,subject,message)
#   movement (in)       active
#   pendant (in)        active
#   phone (out)         text
#   pressure (in)       reading(millibars)
#   smoke (in)          clear, active
#   sms (in)            text
#   sms (out)           text
#   temperature (in)    reading(centigrade)
#   tv (out)            channel_down, channel_set(digits), channel_up,
#                       mute, off, on, volume_down, volume_up
#   vcr (out)           back, channel_down, channel_set(digits),
#                       channel_up, fast_forward, fast_reverse, forward, mute*,
#                       next, off, on, pause, play, previous, record, stop,
#                       volume_down, volume_up
#   washer (out)        off, on
#   window (in)         open, shut
#   wrist (in)          active
#
# Device parameters in the above list are shown in parentheses. Asterisked
# actions may not be available on all devices.

proxy.entities          alert, door, fall, heating, light, movement, phone, sms,
                        speech, temperature
```

See [3] for more detailed information about the SOAP proxy. Note that the SOAP Proxy requires a modified version of the Knopflerfish Axis1 package (see the *README* file in *knopflerfish/osgi/jars/axis-osgi*).

3.4.22 Speech Recogniser

This bundle accepts *speech_in* events with *sender* indicating the sending device and *message* giving the full path to an audio file. Audio can be in most reasonable formats including 8kHz and 16kHz sample rates as well as ADPCM and PCM encoding. The audio is converted into FLAC format using JAVE (http://www.sauronsoftware.it/projects/jave) which in turn is a wrapper for FFmpeg (http://www.ffmpeg.org).

Audio is then sent to the Google Speech Service (Version 2) for recognition. The interface to this service is based on ideas from JARVIS (https://github.com/The-Shadow/java-speech-api). The result is a *receive_message* message with *sender* as the sending device and *message* as the recognised text. By virtue of the Google Speech service, recognition is possible in multiple languages. However this means that, unlike most other bundles, the speech recogniser requires the ACCENT system to have an Internet connection.

The bundle uses the working directory *SpeechRecogniser* located in the ACCENT properties directory. This working directory is created automatically if needed and is used for temporary speech files. The speech recogniser is defined by the property file *SpeechRecogniser.properties*, e.g.:

```
# The owning entity on whose behalf events will be triggered (user@domain):

speech.entity           admin@cs.stir.ac.uk

# Speech API key (see http://www.chromium.org/developers/how-tos/api-keys)

speech.key              key

# Speaker language (<language>-<COUNTRY>, default en-GB)
```

```
speech.locale           en-GB
```

Input to the speech recogniser can be simulated through OSGi. Choose the *Events* tab in Knopflerfish and click the *Send* button. Set something like following, then click the *Send* button.

```
Event Topic      uk/ac/stir/cs/accent/speech_in
sender           nabaztag
mesage           C:/knopflerfish/accent/002185ba6790.wav
```

3.4.23 Tunstall Driver

This bundle is installed in Knopflerfish with start level 8 (so CommAccess is initialised first). It uses the serial port defined by the property file *TunstallDriver.properties*, e.g.:

```
# The mapping translates each sensor signal to trigger parameters (which can be
# omitted from the right). The sensor id is a location code assigned by the PC
# Connect application to a sensor. Policy parameter values are reserved for
# future use. Sensor data is case-sensitive, and the trigger data must match
# policies in respect of case. Spaces are allowed round commas, and "=" must
# be escaped as "\=".
#
#   key:   sensor id,sensor code
#   value: message type,entity name,entity instance,parameter values

# If the signal from a sensor is repeated with the same parameters, it is
# ignored and no event is generated.

# The Tunstall sensors supported are the Bed/Chair Occupancy, Door Switch,
# Medication Dispenser, Personal Trigger and PIR (Standard).

# base unit buttons:

00,HA            click,lifeline,red
00,Ha            click,lifeline,green
00,Hc            click,lifeline,yellow

# wrist alarms:

01,AA            active,pendant,resident1

02,AA            active,wrist_alarm,resident2

# bed sensor:

21,AZ            occupied,bed,lounge
21,BA            free,bed,lounge

# door sensors:

80,AQ            open,door,front
80,AR            shut,door,front

81,AQ            open,door,back
81,AR            shut,door,back

# movement sensors:

41,BH            active,movement,kitchen

51,BH            active,movement,lounge

# medication dispenser

02,CZ            missed,medication
02,JH            taken,medication
```

Input from a wireless device to the policy server can be simulated through OSGi. Choose the *Events* tab in Knopflerfish and click the *Send* button. Set something like following, then click the *Send* button.

```
Event Topic      uk/ac/stir/cs/accent/device_in
user             admin@house.stir.net
arg1             occupied (message type)
arg2             bed (entity name)
arg3             lounge (entity instance)
```

3.4.24 Tuple Server

The tuple space server provides the policy store. Install it in Knopflerfish with start level 7 (before PolicyServer is initialised). This uses IBM TSpaces (http://www.alphaworks.ibm.com/tech/tspaces/download); note that this is available only on an evaluation basis. It listens on port 8200 on the machine where it is started. Port 8201 is typically configured to support a web interface to check tuple space contents. It starts an XML database defined by the property file *TupleServer.properties*, e.g. the following that is adapted from the TSpaces sample configuration file:

```
# The [Server] section contains general specifications for the TSServer.

[Server]

# The port that it listens to for requests.

Port =                          8200

# Default Space options

# If tuple results should be returned in FIFO order

ResultOrderFIFO =               false

# The pathname of the directory used for checkpointing

CheckpointDir =                 C:/usr/local/tspaces/ts-checkpoint

# The interval between dumping the checkpoint data (minutes)

CheckpointInterval =            10.0

# The number of updates before checkpoint is requested (-1 to disable)

checkpointWriteThreshold =      -1

# The interval beween checking for deadlocked threads (seconds)

DeadLockInterval =              15

# The interval between scans for expired tuples (minutes)

ExpireInterval =                15

# The [HTTPServer] section contains parameters for the internal HTTP server

[HTTPServer]

HTTPServerSupport =             true
HttpPort =                      8201

# Turn off the Web Admin support (only use this for development)

HTTPAdminSupport =              false

# The directory where downloadable class files are kept

ClassesDirectory =              ./
```

```
# The [FileStore] section contains parameters for the internal FileStore

[FileStore]

# The directory where files are stored

CacheDir =                    C:/usr/local/tspaces/ts-cache

# The [AccessControl] section contains parameters needed for AccessControl

[AccessControl]

# If false, no access checking will be done

ACVerifierClass =             com.ibm.tspaces.security.AccessControlVerifier

CheckPermissions =            true

# The set of Users and Groups.

AdminUser =                   root
AdminPassword =               ------
AdminGroup =                  AdminGroup

# The list of valid groups

TopGroup =                    Users

[Group-Users]

accent
root

# Subgroups of "Users"

Group AdminGroup

 [Group-AdminGroup]

root

# The [DefaultACL] section sets up the Default Access Control List

[DefaultACL]

Accent                        Read Write
Root                          Read Write Admin
Users                         Read

# The [CreateACL] section sets up the Access Control List that is used
# to control who can create new Spaces.

[CreateACL]

Accent                        Create
root                          Create
```

See [2] for more detailed information about how the policy store is used.

3.4.25 TuxDroid Driver

The TuxDroid bundle should be installed in Knopflerfish at start level 7. It is configure by the property file TuxDroid.properties, e.g.:

```
# The TuxDroid supports a variety of different policy triggers and actions as
```

```
# follows. Alternatives are separated by '|'. Internal trigger or action
# parameters are the same as policy parameters.

# The owning entity on whose behalf events will be triggered (user@domain):

tuxdroid.entity                              admin@cs.stir.ac.uk

# Name of the TuxDroid audio device:

tuxdroid.audio.device                        Speakers (TuxDroid-Audio)

# The interval between reporting average light level (minutes):

tuxdroid.light.interval 5

### Input Triggers ###

# Battery low warning:

battery                                      battery

# Button click on head or wings:

click,head                                   click,head
click,wing,eft|right                         click,wing

# Light reading:

reading,light                                reading,light

### Output Actions ###

# Lights (flash or wink switches light on then off, off/on leaves the light off
# or on until told to change this):

off|on|flash|wink,eyes,left|right|both       off|on|flash|wink,eyes

# Mouth (speak = mouth opens and closes repeatedly):

open|shut|speak,beak                         open|shut|speak,beak

# Eyes:

open|shut|blink,eyes                         open|shut|blink,eyes

# Reset (wings down, eyes open, facing forward, beak closed, lights off):

reset                                        reset

# Body (roughly left/right = 180 degrees, spin left/right = 360 degrees):

rotate,body,left|right|spin_left|spin_right rotate,body

# Wings:

set,wings,up|down|flap_slow|flap_fast        set,wings
```

3.4.26 X10 Driver

The X10 bundle is adapted from code by Michael Wilson. It uses the Java X10 package from Jesse Petersen and others (http://x10.homelinux.org). Install the bundle in Knopflerfish with start level 8 (so CommAccess is initialised first). It uses a serial port defined by the property file *X10Driver.properties*, e.g.:

```
# The port should be the serial port where the X10 computer module can be found
# (appears in Windows Device Manager under Ports as "Prolific USB-to-Serial
# Comm Port"):
```

```
x10.port   COM6

# The following mapping describes a comma-separated list of mappings from
# policy actions to protocol commands. The key data must match policies in
# respect of case, but the value data can be in either case. Spaces can be used
# after commands, and special characters such as "=" must be escaped as "\=".
#
#   key:   message,entity,instance,parameters
#   value: command,address,parameters
#
# One or more messages or commands may be given, separated by "|"; the number of
# these in the key and value must be the same. The instance is optional. The
# parameters can be comma-separated and are optional; if a policy action does
# not match the mapping with its specific parameters, a match is tried without
# the parameters.
#
# Parameters then instance may be omitted from the right (e.g.
# "message,entity,instance" and "message,entity" or "command,address" can be
# used).
#
# Mapping entries can be repeated for the same message, entity and instance but
# with different parameters.

# All X10 devices support "off" and "on" actions. Specialised policy actions are
# as follows:
#
#   X10 Light:          dim

# Standard lamp in lounge

off|on|dim,standard_lamp,lounge      off|on|dim,c2

dim,standard_lamp,lounge,very_dim    dim,c2,20
dim,standard_lamp,lounge,dim         dim,c2,30
dim,standard_lamp,lounge,moderate    dim,c2,50
dim,standard_lamp,lounge,bright      dim,c2,70
dim,standard_lamp,lounge,very_bright dim,c2,85

# Bedside lamp in lounge

off|on|dim,bedside_lamp,lounge       off|on|dim,c3

dim,bedside_lamp,lounge              dim,c3
dim,bedside_lamp,lounge,very_dim     dim,c3,20
dim,bedside_lamp,lounge,dim          dim,c3,30
dim,bedside_lamp,lounge,moderate     dim,c3,50
dim,bedside_lamp,lounge,bright       dim,c3,70
dim,bedside_lamp,lounge,very_bright  dim,c3,85

# Blinds in lounge

open|close|set,blinds,lounge         open|close|set,c1
```

Output to an X10 device from the policy server can be simulated through OSGi. (This requires PolicyAction to be running.) Choose the *Events* tab in Knopflerfish and click the *Send* button. Set something like following, then click the *Send* button.

```
topic           uk/ac/stir/cs/accent/device_out
arg1            on          (message type)
arg2            light       (entity name)
arg3            bathroom    (entity instance)

topic           uk/ac/stir/cs/accent/device_out
arg1            dim         (message type)
```

```
arg2            light       (entity name)
arg3            bathroom    (entity instance)
arg5            25          (parameter values, here the dim percentage)
```

4 Conclusion

This report has described the architecture, installation and configuration of components in the ACCENT policy system. It has been seen that all ACCENT components are bundles deployed in the Knopflerfish OSGi platform (though several of the key components will also run as applications). Most bundles are configured through property files deployed in the Knopflerfish root directory. The components exchange information through the OSGi Event Admin service.

References

[1] Gavin A. Campbell. Overview of Policy-Based Management using POPPET, Technical Report CSM-168, Computing Science and Mathematics, University of Stirling, June 2006.

[2] Stephan Reiff-Marganiec, Kenneth J. Turner, Lynne Blair and Feng Wang. The ACCENT Policy Server, Technical Report CSM-164, Computing Science and Mathematics, University of Stirling, August 2013.

[3] Kenneth J. Turner. Device Services for The Home, in Khalil Drira, Ahmed Hadj Kacem and Mohamed Jmaiel, editors, Proc. 10th Int. Conf. on New Technologies for Distributed Systems, pages 41–48, IEEE Computer Society, Los Alamitos, California, USA, June 2010.

[4] Kenneth J. Turner and Gavin A. Campbell. The ACCENT Policy Wizard, Technical Report CSM-166, Computing Science and Mathematics, University of Stirling, April 2014.

[5] 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, Computing Science and Mathematics, University of Stirling, April 2014.