



# **PiWxRx**

## **A Pi-based weather Alerting System**

## **Installation and operation**

**Written by:**  
**Martin Alcock, M. Sc, VE6VH**

**For software revision: 4.4.1**

## Table of Contents

List of Figures .....	iii
List of Tables .....	iii
Document Revision History.....	iv
Distribution .....	iv
Reference Documents.....	v
Glossary of Terms.....	v
Intellectual Property Notice.....	vi
Disclaimer.....	vi
Release Notes .....	vii
Building the hardware.....	10
Powering the Pi .....	11
Building the HAT EEPROM .....	12
Installing and testing the Software .....	13
Testing the software .....	13
Command line arguments and debugging .....	14
Configuring the software .....	15
System Tag .....	15
Source Tag.....	15
PBX Tag.....	16
Forwarding Tag .....	17
Posting to a Website .....	19
String substitution.....	20
Customizing the decoder .....	21
Starting the service .....	22
Running more than one instance.....	22
Web Server.....	23
Status Page.....	23
SIP Settings.....	24
Audio Source and Codecs .....	24
Forwarding Settings .....	25
PiWxRxWeb.....	26
Setting up the SQL database .....	28
Populating the database .....	28
Configuring PiWxRx.....	31
Navigating the site .....	32
Subscribing to SAME codes.....	33
Sample e-mail .....	34

## List of Figures

Figure 1 PiWxRx Hardware .....	10
Figure 2 Pi Power HAT Schematic .....	11
Figure 3 Web Server status page .....	23
Figure 4 SIP settings .....	24
Figure 5 Audio Source and Codecs.....	24
Figure 6 Forwarding Settings .....	25
Figure 7 SAME Area codes for Jasper National Park.....	29
Figure 8 SAME codes for NY.....	30
Figure 9 Create an account parameters .....	32
Figure 10 Subscribing to SAME codes.....	33
Figure 11 Interpreted Weather Alert Message.....	34

## List of Tables

Table 1 Revision History.....	iv
Table 2 Distribution.....	iv
Table 3 Pi HAT files.....	12
Table 4 Command line switches .....	14
Table 5 Debug mode flags.....	14
Table 6 PiWxRx Configuration XML Tags .....	15
Table 7 System XML Tag Parameters.....	15
Table 8 Source XML tag parameters .....	16
Table 9 PBX XML tag .....	16
Table 10 Forwarding Methods.....	17
Table 11 Forwardng Parameters.....	18
Table 12 Website posting fields.....	19
Table 13 String substitutions .....	20
Table 14 Server login XML parameters.....	27
Table 15 Web Server XML Parameters .....	27
Table 16 SQL Database tables.....	28

## Document Revision History

Date	Rev	Description
<b>November 2022</b>	4.3	Fixed USB bug, added more email posting methods & options
	4.3.1	Fixed software revision on HTML landing page, added originator
	4.3a.2	Added HTML server description
	4.4	Added String substitution and debug sections
<b>May 2023</b>	4.4.1	Added parameters for HTTP and SIP ports

Table 1 Revision History

## Distribution

Rev	Distribution
<b>All</b>	For amateur use only.

Table 2 Distribution

## Reference Documents

- [1] gnu.org, "General Public Licence," [Online]. Available: <https://www.gnu.org/licenses/gpl-3.0.en.html>. [Accessed 25th February 2018].
- [2] National Oceanographic and Atmospheric Administration, "NOAA Weather Radio Alerts," [Online]. Available: [https://www.weather.gov/grb/nwr\\_same](https://www.weather.gov/grb/nwr_same). [Accessed 24 May 2020].
- [3] Network Working Group, "RFC 2543: SIP: Session Initiation Protocol," 2002.

## Glossary of Terms

PBX	Private Branch Exchange. A node in a telephone network that provides connectivity for a series of local extensions to a set of trunks.
VOIP	Voice over Internet Protocol. A system where telephone calls are placed, and audio is exchanged using the Internet Protocol.
NOAA	National Oceanographic and Atmospheric Administration. Originator of weather broadcasts in the US.
SAME	Selective Area message Encoding.
POE	Power over ethernet.

## Intellectual Property Notice

The hardware components and all intellectual property described herein is the exclusive property of Praebius Communications Inc (“the owner”), all rights are reserved.

The owner grants licence to any Amateur for personal or club use, on an as is and where is basis under the condition that its use is for non-commercial activities only, all other usages are strictly prohibited. Terms and conditions are governed by the GNU public licence [1].

No warranty, either express or implied or transfer of rights is granted in this licence and the owner is not liable for any outcome whatsoever arising from such usage.

Copyright © Praebius Communications Inc, all rights reserved.

## Disclaimer

This document is a preliminary release for a product still in development and may be subject to change in future revisions. The software may be subject to unpredictable behaviour without notice. You are advised to keep a can of RAID™ Ant, Roach and Program Bug killer handy. Spray liberally on the affected area when needed.

The following changes were made in Rev 4:

**1. Bulletin Codes moved to SameDB.json file.**

In previous versions, the bulletin codes were hard-coded in the source. They are now in a new JSON stanza 'Bulletins', which is loaded at run time, and is easier to add or revise codes.

**2. Split Site SIP messaging removed**

Previous versions required SIP messaging to support the decoding, which enabled a 'split site' decoder where raw messages could be sent to another instance of the receiver for decoding. Now the message is dispatched according to the selected method instead.

**3. CLI switches removed**

In previous versions the functions of the XML file were duplicated as command line switches. This had been removed and all parameters must now be configured in the XML file.

**4. Beacon message removed**

The beacon message that was sent on startup has been removed, and in turn the messaging stanza has been deprecated. The latitude/longitude is no longer required.

**5. New fields in the forwarding stanza**

The previous station callsign has been replaced by an 'originator' field in the forwarding stanza of the XML file. It is recommended that this reflects the calls sign of the originating station, not the amateur who owns it.

**6. New posting method**

The previous posting method was limited to websites that supported the SOAP protocol. This has been expanded to include a simple HTTP post as well.

**7. Single XML file**

The previous scheme of two different XML files, one for the RTL dongle and a second for a USB device, have been merged into a single file. Examples for different audio sources are contained in the file and can be enabled with a text editor. The documentation has also been improved.

**8. Default configuration**

To aid in testing, the disk is shipped with a file reader enabled and a sample audio file that contains a tornado watch message. This should aid in setup and debugging.

#### **Update Rev 4.1:**

**1. Rework of File reader**

The file reader utility, previously used for testing only, is now able to connect to sound devices managed by ALSA, the Advanced Linux Sound Architecture. It has been tested with both a simple audio dongle and an RA-40 from Master's communications.

**2. Elimination of second runtime library**

The second object library, libpiwxrxUSB.so, has been eliminated. Its purpose was to support devices which sample at 48Ks/sec, to facilitate use of ALSA utility programs. As this functionality is now included in the file reader, this library is no longer necessary.

**3. Found a missing final delimiter in some broadcast messages. Added a test for a non-ascii character at the end of message as an interim fix.**

#### **Update Rev 4.2**

Source code on the distribution disk reflected original release 4.0, Updated code to include new file reader introduced in 4.1.

#### **Update Rev 4.3**

1. Fixed USB audio drop out. Particular to the Raspberry Pi, the USB and ethernet share the same port. Activity on ethernet would cause a buffer overrun condition to occur and the audio to stop. Previously the only recovery method was to restart it, now it recovers and logs a message to the console.
2. Added the ability to supply a custom subject line in the e-mail forwarding mode to the XML setup file. Parameter must be supplied as there is no default. The originator field is appended to the subject line.
3. Fixed bug on e-mail forwarding when no authorization is required. Previously omitting the 'authuser' and 'authpasswd' fields would cause the mailer to crash when not using authorization. These fields are now ignored unless authorization is specified.
4. In the email forwarding mode, the protocol parameter is used to specify the message formatting method. The possible values are listed below, the default is html.
  - 'html' the message content is formatted as 'text/html' with <br> tags for line endings
  - 'plain' the message content is formatted as 'text/plain' with a newline at the line endings

#### **Update Rev 4.3a**

Updated the software revision on the HTML landing page, and added the originating station.



**Update Rev 4.4**

1. Added a string parser to the email subject line to enable message fields to be inserted at will. Subject line is no longer fixed. Docs updated to reflect change. Updated xml configuration file to add agency and originator.
2. Added subject line to dump mode for verification.
3. Added a deemphasis filter to codecs module that can be enabled as a debug mode in the preliminary release. Full release will have an xml parameter for it.

**Update 4.4.1**

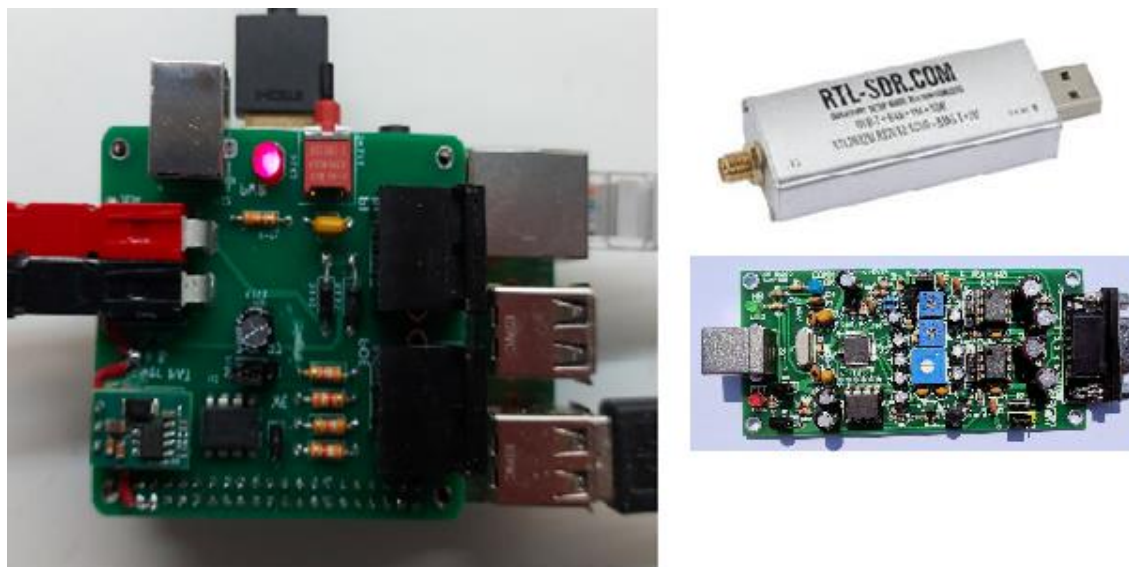
1. Added a new XML parameter to the PBX stanza in the XML file to enable the return port to be specified, enabling more than one instance of the receiver to be realized on the same machine. Previously it was hard coded to 6060, which is now the default if not specified.
2. Added a new command line switch to specify the HTTP port number, default is 8082. Must be unique for each instance.

## Building the hardware

The Pi Weather Receiver (PiWxRx) is a standalone addition to the Meshphone network to receive and decode weather alert messages send by NOAA [2], the National Ocean and Atmospheric Administration. The receiver consists of hardware and software components.

The hardware components are shown in Figure 1 and consist of:

1. A raspberry Pi 2 or 3.
2. Pi Power HAT.
3. An audio source from either an RTL-SDR receiver or USB audio device.
4. A USB hub (recommended)



*Figure 1 PiWxRx Hardware*

The Pi board can be purchased off the shelf from several source, contact the author to obtain a HAT board. The Pi can be powered either from a 12v supply connected to the power pole connectors or using the 802.3af POE standard. In this case, the ethernet cable with the power is connected J103 of the HAT, and the other can be connected to the Pi with a short cable.

It is recommended that an external hub is used to power the USB device, as the current draw from the pi can cause the power supply to droop resulting in poor sensitivity and performance.

Either the SDR or USB radio interface can be used. The software has been tested with several different types of USB SDR, as well as an inexpensive audio dongle, or a more amateur oriented device such as an RA-35, -36 or -40.

## Powering the Pi

The pi power HAT offers a regulator so that a 12v power supply can be used, and power over ethernet (POE) using the 802.11af standard. Figure 2 shows the schematic of the HAT.

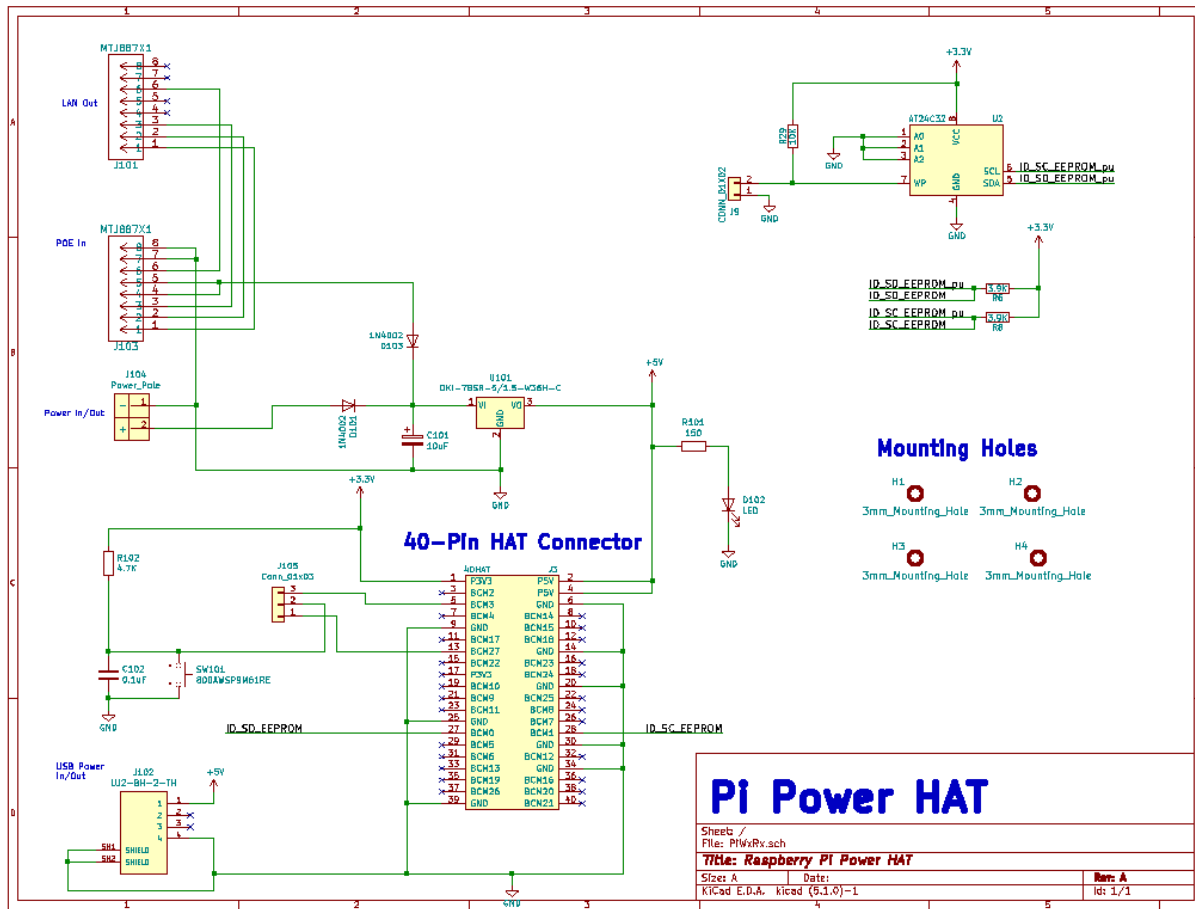


Figure 2 Pi Power HAT Schematic

There are five sections to the HAT hardware:

1. **External supply and regulator.** The 12 supply is connected to a set of Anderson power poles. A switching regulator produces the +5v supply for the pi from this source, the maximum input voltage is 36V.
2. **POE connectors.** Two ethernet connectors are used to connect to a downstream POE source, and one to a the LAN connector on the Pi. The POE circuit supplies the same regulator with a diode mixer.
3. **USB connector.** A type A USB connector that can supply +5V to an external hub.
4. **GPIO input.** A switch can supply a +5v or ground to either GPIO pin 3 or pin 27.
5. **ID EEPROM.** Used to identify the HAT type.

## Building the HAT EEPROM

The EEPROM identifies the HAT device to the raspberry Pi operating system. This is currently not used by the software but may be of use in future versions. When the Pi boots up, it generates five files in /proc/device-tree/hat. The files are shown in Table 3.

File	Contents
<b>name</b>	Name of the device (ie PiWxRx)
<b>product</b>	A description of the product
<b>product_id</b>	A 16 bit number identifying the product
<b>product_ver</b>	A 16 bit number identifying the version
<b>vendor</b>	The name of the vendor

Table 3 Pi HAT files

Building the EEPROM requires three steps:

1. **Build the utilities.** The software can be found in the eeprom directory. Type 'make' to build the utilities.
2. **Build the .eep file.** Run the eepmake utility with the input file 'PiWxRx\_Contents.txt' and output file 'PiWxRx.eep'.
3. **Flash the eeprom.** Insert a jumper into J9 in the board to enable the write, and log in as a super-user. At the console, type the following command:

```
./eepflash.sh -w -f=PiWxRx.eep -f=PiWxRx.eep -t=24C32
```

When flashing is completed, remove the jumper and reboot the pi. The files listed in Table 3 will now be there.

## Installing and testing the Software

The software is available as a pre-built SD card image. Download the image and use WinDiskImager on windows or dd on Linux to write it to an SD card, a 4Gb card is recommended. Insert it into the Pi and apply the power. After it has finished booting you should see the RED led flash as a heartbeat.

There are two ways to communicate with the device for configuration, one is to plug in an HDMI monitor and USB keyboard, and the second is to use SSH to access the device remotely via ethernet. The preconfigured image is set up to obtain an IP address using DHCP. To find the address, connect to your node with a web browser, enter the setup and visit the 'Port Forwarding, DHCP and Services' page.

The login ID is 'pi' and the password is 'PiWxRx'. Once logged in, change to the directory 'piwxrx'. You are now ready to test and configure the software.

### Testing the software

The code is preconfigured to use an audio file as its source for testing purposes. It is recommended to run this first, to ensure that it is working correctly. Then, customize it for your hardware and forwarding scheme. From the command line, run the shell script to start the software as follows:

```
./runpiwx.stdout
```

If any errors occur, they will be shown on the console. To stop the software type <control/C> at any time. Call the extension that you configured and test for audio. If there is none, move the antenna or change the gain.

You should see three decodes of the same message. The software will not terminate, so use the <control/C> mode to exit.

## Command line arguments and debugging

The software contains several debug features that dump messages to the console. This is primarily for development purposes, but can be used to enable some preliminary features as well. The 'piwxrx' files contains the following:

```
export LD_LIBRARY_PATH="/usr/local/lib"
java -cp "./bin:/usr/local/lib/*" PiWxRx -d 800 -X PiWxRx.xml
```

The command line switches and arguments are listed in Table 4

Switch	Argument	purpose
-h	None	Help on the command line arguments
-i	IP Net	Specifies which IP net to use. For systems with more than one NIC
-l	JNI Library	Specifies the name of the JNI library
-n	No load	Does not load library, only runs Java code.
-X	XML File	Specifies the name of the XML configuration file.
-d	Debug flags	Specifies the debug modes
-H	HTTP port	Specifies the port for the built-in HTTP Server. Default is 8082.

Table 4 Command line switches

Note that the XML parameters overwrite everything except the debug flags. This fields is a binary field made up of a combination of bits that can be specified together. Enabling a Debug modes will affect the ability of the receiver to decode messages as it severely degrades the real time performance. The

Flag	Hex	Debug data source
DEBUG_NONE	0x0000	Debugging not enabled.
DEBUG_MSGS	0x0001	Minimal operational messages.
DEBUG_OSC	0x0002	Oscillator values.
DEBUG_LPF	0x0004	Output of the I channel LPF.
DEBUG_DEMOD	0x0008	Demodulator output.
DEBUG_UDP	0x0010	Asterisk PBX UDP information.
DEBUG_WRITE	0x0020	Writes to stdout pipe
DEBUG_BITSHIFT	0x0040	Demodulator byte assembly bit shifter
DEBUG_BYTEOUT	0x0080	Demodulator output.
DEBUG_SYNC	0x0100	Sync (preamble) correlator.
DEBUG_JNI	0x0200	Calls to Java Native Interface
DEBUG_USB	0x0400	USB device reads and status
DEBUG_FEATURE	0x0800	Enables experimental features

Table 5 Debug mode flags

The feature debug is reserved for experimental features. In the Rev 4.4 pre-release it enables the PCM de-emphasis feature.

## Configuring the software

The PIWxRx system is configured with one XML file. There are four stanzas identified by XML tags that need to be configured.

XML Tag	Contents
<b>system</b>	Identifies the system library, and IP network
<b>source</b>	Identifies the audio source
<b>pbx</b>	Identifies the connection to the PBX
<b>forwarding</b>	Specifies the forwarding method for alert messages

Table 6 PiWxRx Configuration XML Tags

### System Tag

```
<system jni="yes" ipnet="10" libname="piwxrx"/>
```

The system stanza determines the run time library which contains the PBX codec and NOAA modem, and the IP network to use:

Parameter	Contents	User modifiable fields
<b>jni</b>	enable/disable audio	Determined by the hardware type
<b>libname</b>	audio run time library	
<b>ipnet</b>	IP network	set to the first octet of the IP network address, normally 10 for the AREDN network

Table 7 System XML Tag Parameters

### Source Tag

```
<source cmdline="/usr/bin/rtl_fm -M fm -f 162.4M -g 38 -"/>
<source cmdline="JNI/filereader -l -f JNI/rx48.raw" />
<source cmdline="JNI/filereader -l -uc 1 -g 0" />
```

The source tag identifies the name of the child process to run to obtain the audio source, and its parameters. Two are supported, GNU radio rtl\_fm for the RTL-SDR dongle, and the filereader for either a disk file or also audio source. The first example is for an SDR dongle, the second is to read a disk file, and the third to use an external sound card device.

Parameter	Contents	User modifiable fields	Explanation
cmdline	USB or test file mode	filereader	name of program
		-l	required
	Test File name	-f xxx.yyy	rx48.raw
	ALSA Device	-ui xxxx yyyy	vendor ID xxxx & Product ID yyyy
		-uc n	sound card number n
	SDR Dongle	rtl_fm	
		-f <frequency in MHz>	164.4M or 164.55M
		-M <mode of operation>	fm
		-g <RF gain>	0-50, tailor to suit
		-	Must be specified
codec	Codec number	0 – G711 $\mu$ Law, 8-G711 ALaw	
afgain	Audio gain in dB	0, 6, 12, or 18.	

Table 8 Source XML tag parameters

## PBX Tag

```
<pbx ext="5284" url="10.66.49.34:5060" secret="ve6vh"
rtpport="12557" msgext="5289"></>
```

The pbx stanza determines the extension configuration:

Parameter	Contents	User modifiable fields
ext	PBX extension number	Determined by PBX configuration
secret	SIP password	
url:port	URL or IP for the PBX and port number	mypbx.local.mesh:5060 or x.y.a.b:5060. 5060 is the normal for asterisk, but it may vary in others <sup>1</sup>
sipport	Sip port number	Port where sip client listens. Default is 6060.
rtpport	port number for RTP audio	normally set to 12557, can be modified if needed
msgext	SIP messaging extension	target extension for SIP messages, if configured.

Table 9 PBX XML tag

<sup>1</sup> The port is optional, the default is 5060 if not specified.



## Forwarding Tag

```
<forwarding
  originator="XLF339"
  database="SameDB.json"
  method="dump"

  serveraddr="10.48.197.216"
  port="8080"

  toaddress="to@messageRx.com"
  replyaddr="noreply@piwxrx.org"
  subjectline="Message from PiWxRx at @O"

  auth="no"
  authuser="mailuser@piwxrx.org"
  authpasswd="s3cretw0rd"

  protocol="http"
  pagename="PIWxRxWeb/alertservice"

  xmlcmd="alertpost.xml"
  webmethod="postsame"

/>
```

The forwarding stanza is used by the message interpreter to determine how to decode a received message and where to forward it. Each receiver has a decoder built in; and it can be configured in one of four modes:

Forwarding method	Decoding	Purpose
<b>dump</b>	none	raw message is dumped to stdout
<b>sip</b>		raw message is sent by SIP messaging to 'msgext'
<b>email</b>	full	decoded message is e-mailed
<b>post</b>	all fields except SAME	partial message is posted to a web site

Table 10 Forwarding Methods

The forwarding parameters takes on different meanings based on the method used:

Method	Parameter	Purpose	Example
<b>all methods</b>	serveraddr	name of the server that will send the message or URL of web page	smtp.myserver.net or http://www.xxx.com/yyyy
	database	name of the SAME JSON database file for message decoding	SameDB.json
	port	port address	specific to forwarding method
	originator	originating station	callsign of the originating station
<b>sip only</b>	msgext	target extension (in PBX stanza)	specifies where to send the sip message
<b>email only</b>	toaddress	Email address of where to send the message	someone@somehere.com
	replyaddress	Address to use if the recipient tries to reply to the message	noreply@somewhere.com
	auth	yes/no	yes if authentication is required by email server
	authuser	username for authentication	Configuration dependent
	authpasswd	password for authentication	
	protocol	Specify message content formatting. Default is html.	html or plain. html uses tags, plain inserts a new line character
<b>post using http</b>	protocol	specifies posting protocol	http
	pagename	Name of server page	page name at the server
<b>post using SOAP</b>	protocol	specifies posting protocol	soap
	pagename	Name of server page	page name at the server
	xmlcmd	Name of file containing SOAP prototype	filename.xml
	webmethod	name of posting method	xxxxx

Table 11 Forwardng Parameters

## Posting to a Website

Two methods of posting are supported, http and the simple object access protocol. In both modes an access is made to a web page that receives a message that is parsed into fields, but the SAME codes are not interpreted. The partially decoded message contains the following:

Field	Contents
<b>Originator</b>	Originating station from the forwarding XML tag
<b>Agency</b>	Name of agency decoded from the local SAME database
<b>Bulletin</b>	Decoded bulletin type, from the SAME database
<b>TimeofIssue</b>	Time issued in UTC
<b>DateofIssue</b>	Date issued in UTC
<b>PurgeTime</b>	Length that bulleting remains in effect
<b>Areas</b>	Undecoded SAME codes separated by spaces

Table 12 Website posting fields

In the HTML mode the specified page name is requested, with the fields in Table 12 as parameters. This is the least secure method, but adequate for closed networks, and is supported by all platforms.

The SOAP mode is intended for more complex websites such as Microsoft's ASP.NET or Apache SOAP. The parameters are specified in a separate XML file which is filled in on the fly.

A sample XML file is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap=http://schemas.xmlsoap.org/soap/envelope/
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <PostSame xmlns="@U">
      <Originator>@O</Originator>
      <Agency>@A</Agency>
      <Bulletin>@B</Bulletin>
      <TimeofIssue>@T</TimeofIssue>
      <DateofIssue>@D</DateofIssue>
      <PurgeTime>@P</PurgeTime>
      <Areas>@R</Areas>
    </PostSame>
  </soap:Body>
</soap:Envelope>
```

## Sample C# Code

A sample of the C# code to receive the above is shown below. In this case the webmethod parameter must be specified as 'PostSame'.

```
[WebService(Namespace="http://www.va6edn.org/services",
            Description="VA6EDN Alert Posting Service")]
public class AlertService : WebService
{
    [WebMethod]
    public string PostSame(string Originator, string Agency,
        string Bulletin, string TimeofIssue, string DateofIssue,
        string PurgeTime, string Areas)
    {
        .. insert processing code here...
    }
}
```

## String substitution

In both the post and e-mail forwarding a string substitution technique is employed to enable parts of the message to be inserted. This is done only in the SOAP XML prototype file, and the email subject line. Substitutions are identified using a 2-byte syntax, the first is always '@', and the second can be one of those indicated in Table 13

Character	Field	SOAP	Subj	Contents
<b>U</b>	URL	Yes	No	URL of the server from the XML configuration file
<b>A</b>	Areas			Undecoded SAME codes separated by spaces
<b>O</b>	Originator		Yes	Originating station from the forwarding XML tag
<b>A</b>	Agency			Name of agency decoded from the local SAME database
<b>B</b>	Bulletin			Decoded bulletin type, from the SAME database
<b>T</b>	TimeofIssue			Time issued in UTC
<b>D</b>	DateofIssue			Date issued in UTC
<b>P</b>	PurgeTime			Length that bulleting remains in effect

Table 13 String substitutions

Note that not all are available in the subject line.

## Customizing the decoder

The decoder uses the JSON file specified in the forwarding section of the XML file. The database contains five fields as shown in the following example:

```
{
  "Agencies": [
    [ "EC/GC/CA", "Environment Canada"],
    [ "NWS", "The National Weather Service" ]
  ],
  "DateOffset": 1,
  "HourOffset": 7,
  "Bulletins" : [
    [ "BZW", "Blizzard warning" ],
    [ "DSW", "Dust storm warning" ],
    [ "FSW", "Flash freeze warning" ],
    [ "FZW", "Frost warning" ],
    [ "HUA", "Hurricane watch" ]
  ],
  "Locations" : [
    [ "071100", "Jasper National Park"],
    [ "071110", "Jasper Nat. Park near Pocahontas"],
    [ "071120", "Jasper Nat. Park near Jasper"],
    [ "071130", "Jasper Nat. Park near and south of Sunwapta Falls"]
  ]
}
```

Field	Contents	Default
<b>Agencies</b>	An array of abbreviations and agency names	none
<b>DateOffset</b>	An offset to be added when calculating the effective date	US – 0; Canada – 1
<b>HourOffset</b>	Number of hours before GMT	1
<b>Bulletins</b>	A 3-character designator for the bulleting type	none
<b>Location</b>	An array of SAME codes and area names.	

## Starting the service

Once the receiver is running correctly, you can set it up to be invoked automatically at start up. To enable it, enter the following:

```
sudo systemctl enable piwxrx
```

Then reboot the pi, the service should be running. You can check it with:

```
sudo systemctl status piwxrx
```

## Running more than one instance

More than one instance can be run on a single machine as of release 4.4.1. There are three parameters that need special attention, the SIP, RTP and HTTP port numbers. For the first instance, these can be defaulted. Subsequent instances MUST have their own XML file, primarily as the PBX extension will be different.

In the XML file, the sipport and rtpport parameters in the PBX stanza must be specified (see Table 9). The default for the SIP port is 6060, subsequent instances can use 6061, 6062, etc. The default for the RTP port is 12557, any other value can be specified, the simplest is just to add one.

On the command line, the HTTP port number is specified by the -H (see Table 4). The default is 8082, subsequent instances can use 8083, 8084, etc. These are not cast in stone, and if they are already in use elsewhere it may be specified for the first instance as well.

## Web Server

The software has a web server that responds to specified port (default 8082) at the IP address of the receiver. It contains four pages, which are not dynamically updated:

1. Status page
2. SIP settings and status
3. Settings for the audio source and codecs.
4. Forwarding settings.

Changes can be made to a running receiver, but they are temporary and are overwritten at the next restart. For permanent changes, please update the PiWxRx.xml file.

### Status Page

Figure 3 illustrates the status page.

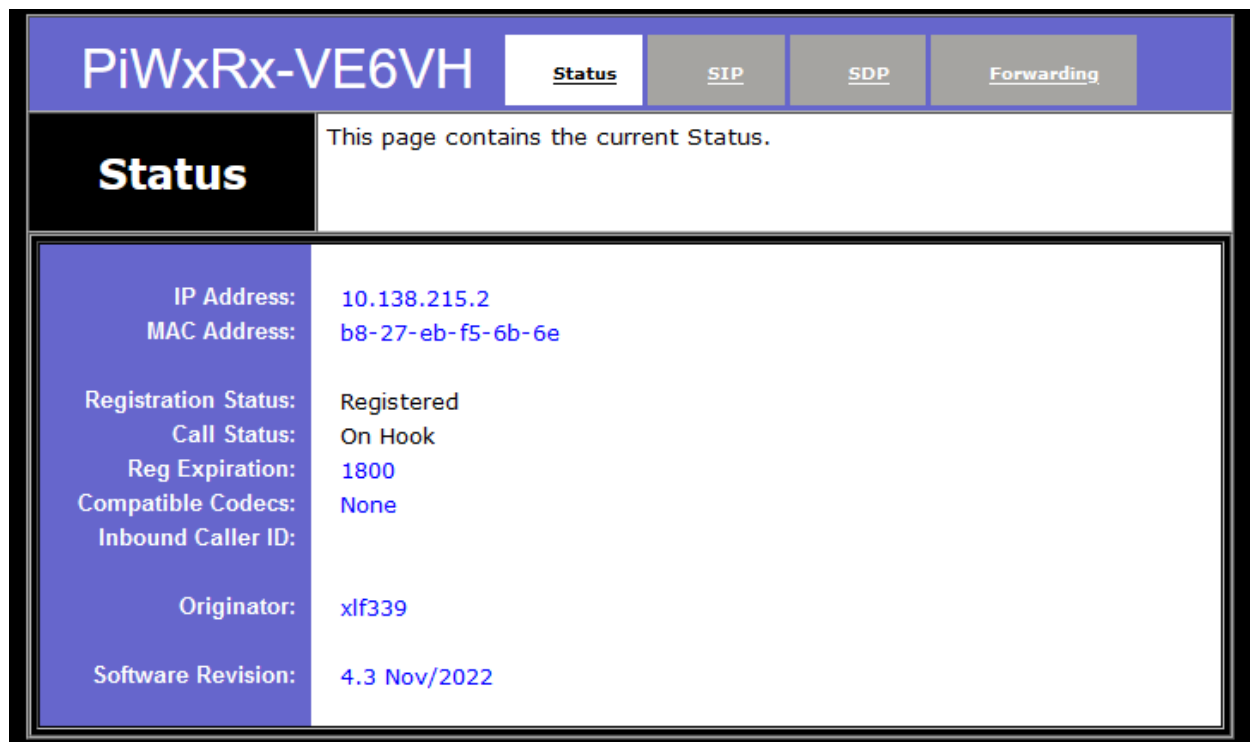


Figure 3 Web Server status page

The fields are relatively straightforward; The registration and call status reflect the current inbound call, if any. To update it the page must be refreshed with an F5.

The originator field is taken directly from the xml file.

## SIP Settings

Figure 4 illustrates the SIP settings.

PiWxRx-VE6VH	
<a href="#">Status</a> <b><a href="#">SIP</a></b> <a href="#">SDP</a> <a href="#">Forwarding</a>	
<b>SIP</b>	This page shows the SIP setup parameters
PBX Extension:	5184
PBX URL:	ve6vh-pbx.local.mesh:5060
Registration Status:	Registered
Call Status:	On Hook
Reg Expiration:	1800
Compatible Codecs:	None
Inbound Caller ID:	

Figure 4 SIP settings

The PBX settings indicate where the receiver is registering, and the extension number. The other settings are the same as the status page.

## Audio Source and Codecs

Figure 5 illustrates the audio source and codecs.

PiWxRx-VE6VH	
<a href="#">Status</a> <a href="#">SIP</a> <b><a href="#">SDP</a></b> <a href="#">Forwarding</a>	
<b>SDP/RTP</b>	This page is for the SDP/RTP protocol parameters
Source Command Line:	/usr/bin/rtl_fm -M fm -f 162.4M -g 30 -
Codec Type:	G711 uLaw <input type="button" value="Select"/>
Audio Gain:	0 dB Gain <input type="button" value="Select"/>
<input type="button" value="Submit"/> <input type="button" value="Cancel"/>	

Figure 5 Audio Source and Codecs



The audio source reflects the command line used to establish the process that generates the audio. In this case, the RTL dongle is being used on 162.4MHz with an RF gain of 30 dB.

The codec type and audio gain can be modified from the drop-down lists, but they will not take effect until the next inbound call.

## Forwarding Settings

Figure 6 illustrates the forwarding settings.

PiWxRx-VE6VH	
<div> <div>Status</div> <div>SIP</div> <div>SDP</div> <div><b>Forwarding</b></div> </div>	
<b>Forward</b>	This page sets up the forwarding method
Forwarding:	<input type="radio"/> Disabled <input checked="" type="radio"/> Enabled
Forwarding Method:	post
Protocol:	http
Server Address:	10.48.197.216
Server Port:	8080
To Address:	to@messageRx.com
Reply Address:	noreply@piwxrx.org
Authentication:	<input checked="" type="radio"/> Not Required <input type="radio"/> Required
User name:	mailuser@piwxrx.org

Figure 6 Forwarding Settings

These are for display only and cannot be modified. In this case the receiver is posting using the http protocol to site at the server address and port shown.

The remaining fields pertain to the e-mail forwarding method only. For more information see the section 'Forwarding Tag' in the xml command file description.

## PiWxRxWeb

PiWxRxWeb is a website that responds to an http post from a receiver and writes the data to a SQL database. Users can setup an account on the site to receive alerts by SAME code, which are delivered by email. The web server installation requires three components:

1. Apache web server
2. Apache Tomcat java web server
3. MySQL or MariaDB SQL database server

The first step is to install all the software. To test that the web servers are functional, try the following:

localhost:80	Verifies operation of the Apache server
localhost:8080	Verifies operation of the Tomcat server

Following the installation of the software, two additional steps are required:

1. Setup server.xml to find the database and parameters for e-mail.
2. Setup the SQL tables and populate them.

### Server.XML

This file requires two stanzas that need to be customized. The first identifies the database and table that contains the user login information. The fields to be modified are listed in Table 14

```
<!-- this realm handles the SQL database -
  <Realm className="org.apache.catalina.realm.JDBCRealm"
    connectionURL="jdbc:mysql://10.48.197.216:3306/piwxrx?user=user&
    ;password=pwd&useSSL=false&serverTimezone=UTC"
      driverName="com.mysql.cj.jdbc.Driver"
      roleNameCol="role"
      userCredCol="password"
      userNameCol="callsign"
      userRoleTable="roles"
      userTable="users">
        <CredentialHandler algorithm="MD5"

        className="org.apache.catalina.realm.MessageDigestCredentialHandl
er"/>
      </Realm>
```

Field	Content
Server URL	URL of the SQL server and port. In the example it is 10.48.197.216
user	Login user name to the database
password	Login password to the database

Table 14 Server login XML parameters

The second contains operational parameters for the web server. All fields need to be customized.

```
<Context docBase="PIWxRxWeb" path="/PIWxRxWeb" reloadable="true"
    source="org.eclipse.jst.jee.server:PIWxRxWeb">
<Environment name="PiWxRxSQLurl" type="java.lang.String"
    value="jdbc:mysql://10.48.197.216:3306/piwxrx?user=user&passw
    ord=pwd&useSSL=false&serverTimezone=UTC"/>
<Environment name="sipportal" type="java.lang.String"
    value="sipportal@va6edn-server.local.mesh"/>
<Environment name="repladdr" type="java.lang.String"
    value="noreply@va6edn-server.local.mesh"/>
<Environment name="smtphost" type="java.lang.String"
    value="va6edn-server.local.mesh"/>
<Environment name="smtpport" type="java.lang.String" value="25"/>
<Environment name="usesmtpauth" type="java.lang.Boolean"
    value="false"/>
<Environment name="mailuser" type="java.lang.String" value="user"/>
<Environment name="mailpasswd" type="java.lang.String"
    value="passwd"/>
</Context>
```

Field	Content
<b>PiWxRxSQLurl</b>	URL of the database, login user name and password. Same as Realm.
<b>sipportal</b>	Email address of the sip portal
<b>repladdr</b>	Content of the reply address field on all e-mails
<b>smtphost</b>	Name of the SMTP server
<b>smtpport</b>	Port address of the SMTP server
<b>usedmtpauth</b>	True if authorization is required, false otherwise
<b>mailuser</b>	User name for authorization
<b>mailpasswd</b>	Password for authorization

Table 15 Web Server XML Parameters

Make the appropriate changes to the file, copy it to the /etc/tomcat/conf directory, and then copy the web archive, PIWxRxWeb.war, to the /etc/tomcat/webapps directory.

When all the configuration is complete, restart the tomcat service, and it will unpack the java files and start the website. It can be invoked by:

Localhost:8080/PIWxRxWeb

Note that any time the server or service is restarted, tomcat will unpack the archive file. If you make changes that are meant to be permanent, then modify this statement to:

```
<Host appBase="webapps" autoDeploy="true" name="localhost"
unpackWARs="false">
```

## Setting up the SQL database

There are 7 tables that need to be setup on the SQL Database, this task should be undertaken before starting the web server.

Table	Populated by	Contents
<b>areanames</b>	User	4- digit SAME codes and area names
<b>location</b>		6-digit SAME codes and location names
<b>receivers</b>		Receivers that are posting to PiWxRxWeb
<b>samemessages</b>	Receivers	Messages posted by receivers
<b>subscriptions</b>	Subscribers	Callsigns of subscribers and areas of interest
<b>users</b>		Users on the system
<b>roles</b>		Roles for users

Table 16 SQL Database tables

If you have an appropriate tool, the file piwxrxWebSQL.sql will create all the files. If not, they will have to be created individually.

## Populating the database

The three user tables show in Table 16 are part of the message decoding system and have to be populated before the system will operate. The area names table contains an abbreviated version of the same code, specifying a larger area that can encompass more than one code.

A comprehensive list for all areas does not exist, but this can be built from several web sites.

For example, the map in Figure 7 shows area codes for Jasper National Park, and Figure 8 shows area codes for Albany, NY.

## PiWxRx weather alerting system installation and operation

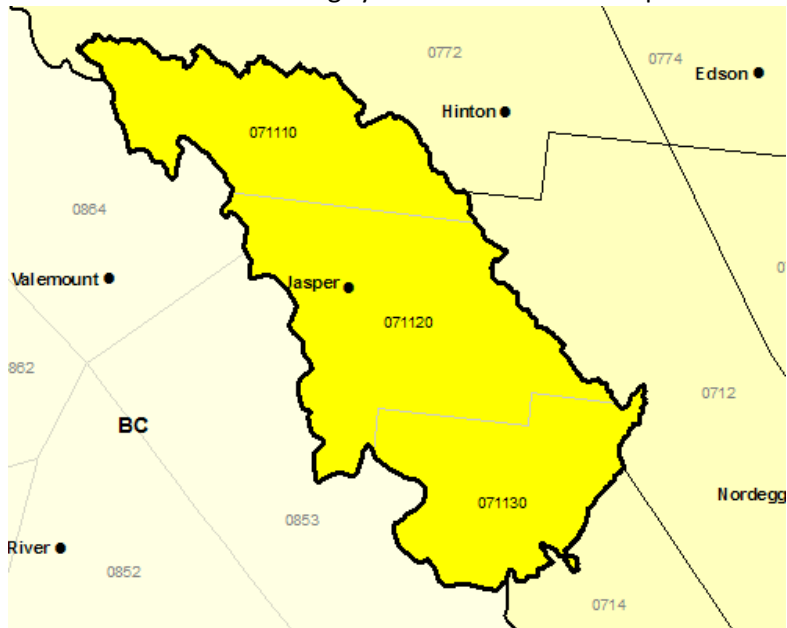


Figure 7 SAME Area codes for Jasper National Park

The area names table will have one entry for all the regions:

AreaCode	AreaName
0711	Jasper National Park

But the location codes table subdivides this into the regions on the map, with a placeholder for messages that apply to the entire area.

LocationCode	LocationName
071100	Jasper National Park
071110	Jasper Nat. Park near Pocahontas
071120	Jasper Nat. Park near Jasper
071130	Jasper Nat. Park near and south of Sunwapta Falls

In the state of New York all codes begin with 0360. Therefore, the area names table will have one entry for all, and the location codes table will contain the full 6-digit codes.

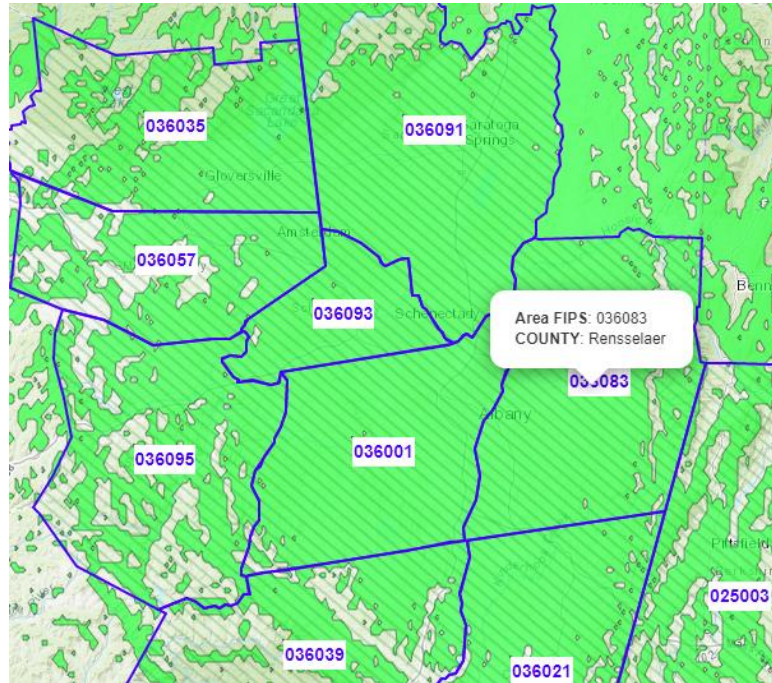


Figure 8 SAME codes for NY

036001	Albany
036003	Buffalo
036003	Call Hill
036005	New York City
036007	Binghamton
036007	Elmira
036007	Honesdale

The receivers table is also hand-populated, it contains data about the callsign, location, frequency and lat/long of the receiver.

VDC816	Brooks	162.4	50.53416667	-111.9158333
XLF339	Calgary	162.4	51.06027778	-114.1702778
XOF962	Cooking Lake	162.475	53.36111111	-112.9025
VBX254	Crowsnest Pass	162.55	49.53166667	-114.1936111

Positive latitudes are degrees N, negative longitudes are degrees W.

## Configuring PiWxRx

To post to the site, PiWxRx needs to be configured with the server, web page and forwarding parameters. Any number of receivers can post to the site as each is treated separately.

The following fields must be setup in PiWxRx.xml:

```
<forwarding
  originator="xlf339"           ; unique between receivers
  database="SameDB.json"       ; name of JSON database
  method="post"                ; use posting method

  serveraddr="10.48.197.216"    ; IP address of your web server
  port="8080"                  ; Apache tomcat port

  toaddress="to@messageRx.com" ; mail fields are not used, but

  replyaddr="noreply@piwxrx.org" ; leave alone for now
  auth="no"
  authuser="mailuser@piwxrx.org"
  authpasswd="s3cretw0rd"

  protocol="http"              ; must be set as is
  pagename="PIWxRxWeb/alertservice" ;

  xmlcmd="alertpost.xml"       ; only used by SOAP
  webmethod="postsame"         ; leave alone for now

/>
```

## Navigating the site


The main landing page of the site is located at <server name>:8080/PIWxRxWeb. This page offers a link to the login page, or to set up an account.

## Welcome to the PiWxRx Weather alert site

**If you have already set up an account**, then please [log in](#) using your user name and password.

If you have not yet set one up, please visit the [Create an account](#) page first.

If you have an account, click the link and log in with your user name and password. If not, click the create link and enter the fields shown in Figure 9.

A screenshot of a web form titled "Please Enter your details:". The form contains six input fields stacked vertically: "Callsign:", "Password:", "First Name:", "Last Name:", "Email Address:", and "Meshphone number:". Below the last field is a button labeled "Create Account".

Please Enter your details:	
Callsign:	<input type="text"/>
Password:	<input type="password"/>
First Name:	<input type="text"/>
Last Name:	<input type="text"/>
Email Address:	<input type="text"/>
Meshphone number:	<input type="text"/>
<input type="button" value="Create Account"/>	

Figure 9 Create an account parameters

Fill in the fields above. Meshphone numbers should be included, but are restricted to the local PBX, and if you plan to use SIP messaging, you must have the email to SIP portal installed, and your PBX must support messaging. As of this time, sending an SIP message to a remote PBX is not supported.

Once logged in, you can navigate to one of four pages:

1. **Alerts.** View up to the most recent 10 alert messages from receivers on the system.
2. **Setup.** Set up the SAME codes of interest, and where to send the messages.
3. **Manage Account.** Change your password or manage your subscriber record.
4. **Log Out.** Log out of the website.



The subscription setup page on the site enables SAME codes to be added as required; the first 4 digits of the area are selected, which will include all subdivisions of this area as well. A sample of a configuration is shown in Figure 10.

## Subscribing to SAME codes

The setup page is invoked from the main menu of the website after logging in, as show in Figure 10. Four items need to be specified:

1. Where to send the message, e-mail or phone<sup>2</sup>, or both.
2. Which SAME codes are to be watched.
3. Whether to include the bulletin in the email message subject
4. Whether to use plain text for e-mail messages

## SAME Area code Subscription

Please select how you would like to be alerted:

☒ Send a copy of the alert message to my meshmail  
☐ Do not include bulletin in subject line  
☐ Send email in plain text format  
☒ Send a SIP message to my meshphone

SAME codes to which you are subscribed.

Same Code	Name	Remove
0714	Banff National Park	<a href="#">Remove</a>
0724	City of Calgary	<a href="#">Remove</a>
0725	Okotoks - High River - Claresholm	<a href="#">Remove</a>

Available codes in your area.

Same Code	Name	Add
0711	Jasper National Park	<a href="#">Add</a>
0712	Nordeg - Forestry Trunk Road Highway 734	<a href="#">Add</a>
0713	Rocky Mountain House - Caroline	<a href="#">Add</a>
0715	Kananaskis - Canmore	<a href="#">Add</a>
0721	Red Deer - Ponoka - Innisfail - Stettler	<a href="#">Add</a>
0722	Airdrie - Cochrane - Olds - Sunde	<a href="#">Add</a>
0723	Drumheller - Three Hills	<a href="#">Add</a>

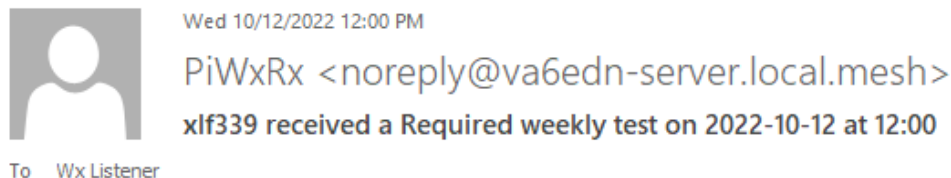
Figure 10 Subscribing to SAME codes

<sup>2</sup> Requires a separate e-mail to SIP portal

## Sample e-mail

Figure 11 shows a sample of the fully decoded message sent as an e-mail message, including the bulletin in the subject line, SMS messages use the subject line only.

An option is provided to shorten the subject line, when checked the subject does not contain the decoded bulletin for e-mail, however the SMS message remains the same.



Environment Canada has issued a Required weekly test bulletin at 11:00 local time on 2022/10/12  
For the following areas:

Kananaskis - Canmore  
M.D. of Bighorn near Canmore Exshaw and Ghost Lake  
Kananaskis Improvement District near Kananaskis Village  
Airdrie - Cochrane - Olds - Sundre  
Rocky View Co. near Airdrie and Crossfield  
Rocky View Co. near Bottrel and Madden  
Rocky View Co. near Cochrane  
Drumheller - Three Hills  
Rocky View Co. near Irricana Beiseker and Kathym  
City of Calgary  
Okotoks - High River - Claresholm  
Rocky View Co. near Bragg Creek and Tsuu Tina Res.  
M.D. of Foothills near Priddis and Brown-Lowery Prov. Park  
M.D. of Foothills near Turner Valley and Black Diamond  
M.D. of Foothills near Okotoks and De Winton  
Brooks - Strathmore - Vulcan  
Rocky View Co. near Chestermere Dalroy and Dalemead  
Wheatland Co. near Strathmore Lyalta and Carseland

for the next 01:00 hours

*Figure 11 Interpreted Weather Alert Message*