EASTERN REGION TECHNICAL ATTACHMENT
NO. 2006-02
December 2006

HOW TO BROADCAST SELECTED NOAA WEATHER RADIO PRODUCTS OVER THE INTERNET

Dan Hagarty
NOAA/National Weather Service Office
Mount Holly, New Jersey

Abstract

This paper discusses how the Weather Forecast Office Philadelphia/Mount Holly NJ deploys and presents selected audio products from the NOAA Weather Radio broadcast suite to web site end users. It provides process definition and discusses procedures that can be used by other forecast offices to deploy a similar site.

1. Introduction

Public expectations have increased as the National Oceanic and Atmospheric Administration (NOAA) National Weather Service Internet sites have become more readily available and capable. The public would like access to all of the products generated by a forecast office online, including NOAA Weather Radio (NWR) broadcasts. This document discusses one method that a forecast office can use to provide select NWR broadcasts to the public via the Internet.

2. Background

Audio files are normally distributed over the Internet as either streaming audio or single file transfer that download audio files to the end user's computer. Streaming audio can be thought of as a pipeline between the Internet user and the broadcast source. This pipeline provides continuous broadcast to the end user, similar to the way NOAA Weather Radio broadcasts are issued now. The user turns it on and listens.

Single file transfer is a simple "request and respond" technique. The user requests a file and the web server makes the file available for download. Once downloaded, the audio file can be "played" on the user's computer.

Providing streaming audio of the NOAA Weather Radio broadcasts to the public could severely impact the current suite of products available from office web sites. Increased bandwidth and processing capability would need to be added to offset this impact; and that is cost prohibitive.

Single file transfer puts much less stress on the web server and network bandwidth than streaming audio. A drawback to this technique is that the user must know which file they want to download and how it should be played. A web page interface can help the end user make this selection by providing a list of available files. The user can select the file that they want to hear and the web interface will trigger the program to play the audio file.

The programs discussed in this document are based on the system that the Weather

Forecast Office Philadelphia/ Mount Holly, NJ (WFO PHI) uses to provide NWR audio files to the public via the office web page. WFO PHI started using this system in the fall of 2005.

These programs will be referred to as *NWRWaves2web*. *NWRWaves2web* uses the single file transfer technique and has been designed to make optimal use of available resources and to have minimal impact on web server bandwidth, storage and processing capability. It uses software that is readily available on the currently deployed versions of the Advanced Weather Information Processing System (AWIPS) and the Console Replacement System (CRS).

The web site location of WFO PHI's version of *NWRWaves2web* is: http://www.erh.noaa.gov/phi/wxradio/nwrweb.html

3. Overview of NWRWaves2web Process

The current release of the Voice Improvement Program (VIP) has tools that allow a forecast office to store a broadcast product in either .wav or .mp3 file format. *NWRWaves2web* uses the mp3 format because it is an accepted standard for exchanging compressed audio files over the Internet and it is readily supported by most personal computers. VIP can be configured so that all voice-ready products can be transferred to the Local Data And Dissemination (LDAD) server automatically. LDAD can then filter the product list and only send the required files to the web server according to an established schedule.

WFO PHI limits the number of products available via *NWRWaves2web* to five or six products for each of the WFO transmitters. The products selected by WFO PHI are issued at regular intervals each day. Each product is overwritten when a new version is transferred to the web server. Versioning support and file cleanup (purging) does not have to be done using this technique

NWRWaves2web is basically an extension to the current set of processes that put AWIPS generated products on NWR (AWIPS->CRS->VIP->NWR)[Fig. 1]. Certain products that are generated in AWIPS are programmed to go to NWR. They get to the radio via CRS and VIP. The *NWRWaves2web* process takes the files generated by VIP and deploys them to the web. Once there they can be accessed via a web page.

4. Procedures to implement NWRWaves2web

Assumptions

The person configuring the system must have the appropriate privileges on AWIPS and CRS. They must also be able to create and deploy the required web pages to the office web site. The person who carries out the procedures listed below is assumed to have the following knowledge:

1. Shell scripting in the AWIPS environment
2. Cron job modification on LDAD
3. VIP option setting
4. Web page design and deployment for the office web site
5. Some familiarity with JavaScript and Cascading Style Sheets

A compressed version of all of these files is available for download at:
http://www.erh.noaa.gov/phi/ito/NWRWaves2web.zip

Procedures

Preliminary procedures

Some preliminary work must be done before implementing *NWRWaves2web*.

1. Decide how to transmit audio files to the web server
   - A link between the VIP computer and LDAD must be available if you decide to use LDAD for deployment of the audio files to the web server. WFO PHI uses a samba link for this. This step is not necessary if the WFO uses another technique to transmit the files to the web server.
2. Define the list of products that will be available on the web site
   - This list must be defined for each transmitter.
3. Create list of audio files for selected products.
   - This list is the filenames of the audio files that will be made available on the web site.

Main Procedures

After insuring all 3 preliminary procedures are complete, there are 3 steps to implement *NWRWaves2web* (Fig. 2).

1. Configure VIP to send mp3 files to LDAD

Select “System Settings” from the Main VIP Interface on the 5MP/VIP console. Select “Options”.
Set the “Audio FTP Configuration” as follows:

a. Check the “On” box
b. Fill in your Username WFO xxx uses “crs”, where xxx is the 3-letter station identifier
c. Enter your Password
d. Set the IP Address of the computer that the files will be transferred to.

e. Put the directory location on the destination computer in the Upload Directory edit box..

2. Configure LDAD to send the files to the web server

WFO PHI uses the following cron tab for user ldad on ls1 to schedule product transmission to the web server. It will need to be customized to support your site requirements. The file /ldad/bin/send_nwr2web.tcl is used in the crontab to transfer the audio files from LDAD to the web server. The full listing for the /ldad/bin/send_nwr2web.tcl is provided in Appendix A. That script will also need to be modified to support local requirements.

```
# Nwr files to the web
23 * * * * csh -c '/ldad/bin/send_nwr2web.tcl HOURLY > /ldad/bin/send_nwr2web.log'
45 6 * * * csh -c '/ldad/bin/send_nwr2web.tcl CLIMATE > /ldad/bin/send_nwr2web.log'
15 * * * * csh -c '/ldad/bin/send_nwr2web.tcl COASTAL > /ldad/bin/send_nwr2web.log'
30 * * * * csh -c '/ldad/bin/send_nwr2web.tcl RIVER > /ldad/bin/send_nwr2web.log'
10 3,7,11,15 * * * csh -c '/ldad/bin/send_nwr2web.tcl RIVER > /ldad/bin/send_nwr2web.log'
35 * * * * csh -c '/ldad/bin/send_nwr2web.tcl OFFSHORE > /ldad/bin/send_nwr2web.log'
```

3. Build the web page interface

The web page interface must allow the web site user to select and play the audio file of their choosing. A working example of the html code to generate this interface is discussed in Appendix B. The key parts of this web page are:

- The background image that displays the CWA and transmitter locations
- The html image map that triggers the menu system
- The JavaScript code that activates the file transfer
- The Cascading Style Sheet (css) that defines element appearance and placement

The image map referenced by this tag (usemap=#nwrmap") triggers the drop down menus. This image map must be customized by each WFO to reflect the transmitters for their CWA. The shapes referenced in the image map refer to the transmitter locations on the image.

The following is a code segment from the WFO PHI implementation. The call to the JavaScript function showMenu() must be changed to indicate the menu that will be displayed for each transmitter. The menu options are defined in the file getMenuItems.js (see Appendix D). showMenu() is defined in the file wxradioweb.js (see Appendix E).

```
<map id="nwrmap" name="nwrmap">
<AREA alt="Allentown, PA Transmitter" onClick="showMenu(this, event, miNW1, '250px')"
SHAPE="circle" COORDS="229,200,10" onmouseover="changeCursor('pointer');"
onmouseout="changeCursor('auto');">
…
Put additional transmitters here
…
</map>
```

The following lines of code are required to be in the html file and should be placed in the <head></head> section. They are instructions to include the JavaScript files in the web page and to accommodate users that have JavaScript disabled. The `<noscript>` tag should keep the page Section 508 compliant. While this implementation does not define how Section 508 is supported, it does provide a place to trigger an alternate file. The css file controls the appearance and positioning of the drop down menus and message boxes (see Appendix C).

```
<link href="css/wxradioweb.css" rel="STYLESHEET" type="text/css" />
<!-- CONFIGURE THE MP3 FILES TO BE PLAYED IN THE SCRIPTS/GETMENUITEMS.JS FILE -->
<script src="scripts/getMenuItems.js" language="JavaScript" type="text/javascript"></script>
<script src="scripts/wxradioweb.js"   language="JavaScript" type="text/javascript"></script>
<script src="scripts/mp3popup.js"     language="JavaScript" type="text/javascript"></script>
<noscript>
<h4 align="center">This web site will not work properly with your browser because Javascript is
not enabled.</h1>
</noscript>
```

5. Conclusion

NWRWaves2web represents one solution for posting NWR audio files to the web. It has been received well by WFO PHI web site users and could be used to standardize other office web sites.

The architecture discussed here does not limit the number or type of products that can be issued. This implementation provides a limited number of products because the author felt that it would be best to start out small to minimize impact to an already stressed web server. As web server processing capability and network bandwidth availability increase, the configuration files used for this implementation can easily be modified to support more products.

The time that it will take an office to implement *NWRWaves2web* for their office web site will vary. The two most time consuming parts are generating the map and testing. Several WFOs have offered to help generate maps for offices that either don’t have them, or don’t have the expertise to build them.

Modifying the support files should not take more than a few hours. Someone familiar with UNIX administration, html and JavaScript should be able to make the modifications and additions to their web site rather quickly once familiar with the code.

Testing should be done over a period of a few days to ensure that all of the processes are running correctly.

Acknowledgments

Many of the ideas used in this design came from Greg Lewis, Information Technology Officer, Columbia, SC and Fortune Vilcko at Eastern Regional Headquarters. Joe Byerly, the Electronics System Analyst at WFO PHI, provided the CRS/VIP setup to generate the mp3 files. He also setup the samba share on LDAD. Mike Gorse, the NWR focal point at WFO PHI, provided input regarding customer requirements and product scheduling. Ron Holmes, Information Technology Officer, State College, PA and Heather Hauser, Applications Development Meteorologist, Eastern Region Headquarters, reviewed the initial version of this document and provided important feedback to the author that has been included in the final version.

References

Some of the JavaScript code used by *NWRWaves2web* was pulled from public domain code repositories and tutorials available at the following web sites: w3schools.com, devguru.com and dynamicdrive.com.

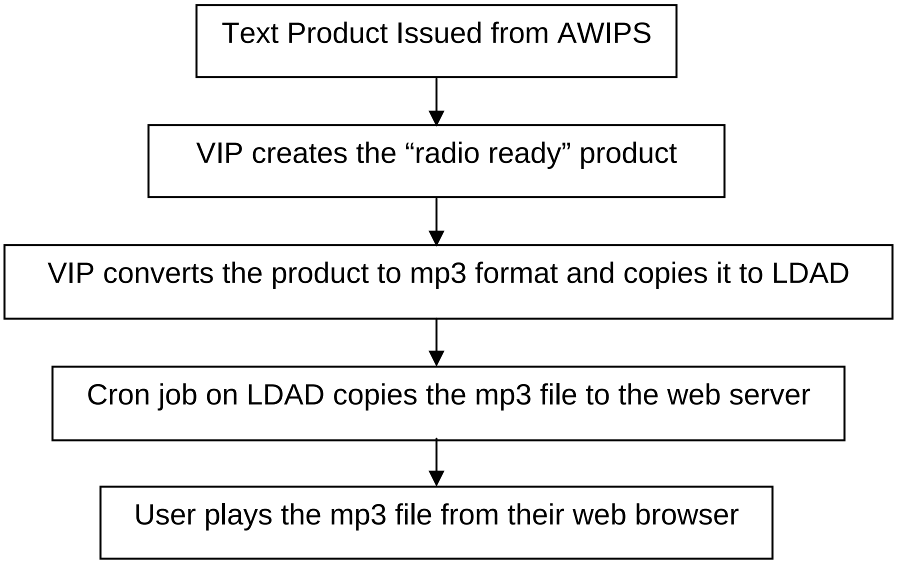
Text Product Issued from AWIPS
VIP creates the “radio ready” product
VIP converts the product to mp3 format and copies it to LDAD
Cron job on LDAD copies the mp3 file to the web server
User plays the mp3 file from their web browser

Figure 1. Data flow diagram of NWRWaves2web process

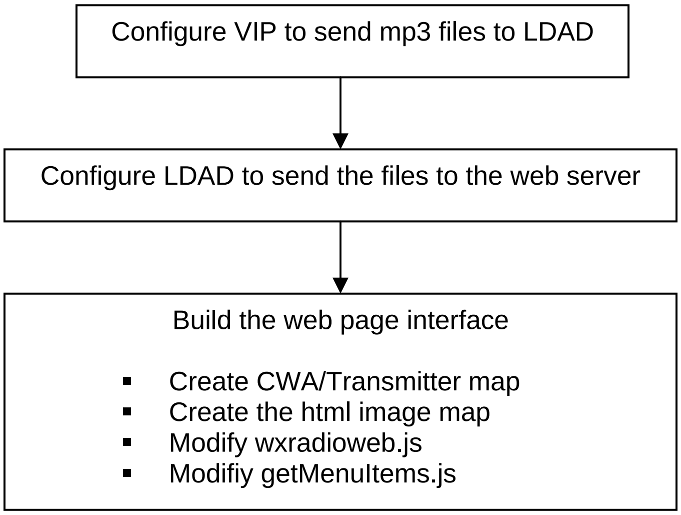
Configure VIP to send mp3 files to LDAD
Configure LDAD to send the files to the web server
Build the web page interface
Create CWA/Transmitter map
Create the html image map
Modify wxradioweb.js
Modifiy getMenuItems.js

Figure 2. Step to Implementing *NWRWaves2web*

Appendix A: – send_nwr2web.tcl

WFO PHI uses the following script to transfer the audio files from LDAD to the web server. The script will need to be customized to support your site requirements (see `# Site specific vars`). The filename is /ldad/bin/send_nwr2web.tcl and is called by the LDAD cron service for user ldad.

```
#!/usr/local/expect-5.32/bin/expect
##########################################################
#
# send_nwr2web.tcl
#
# Purpose:
#   Sends the nwr mp3 files to the web server
#
# Author:
#   Dan Hagarty, ITO, WFOPHI
#
# Requirements:
#   The audio files that are generated by CRS need to be
#   copied to ldad. The Philadelphia/Mount Holly office does
#   this by using a samba directory that CRS and ldad share.
#   (see sourceDir below)
#
# History:
#   version 1.0 -- June 20, 2005 -- First cut at the program
#   version 1.1 -- June 20, 2005 -- Added argument list
#   version 1.2 -- July 6, 2005  -- Deletes old files from the server
#   version 1.3 -- July 7, 2005  -- Improved program control structures, completed arg list
set VERSION 1.3
#
###########################################################

puts stdout "*** Starting $argv0 at: [clock format [clock seconds]]"
set errs 0
#**********************************************************
# Make sure the proper arguments where passed in
#**********************************************************
lappend type_of_send "HOURLY"
lappend type_of_send "CLIMATE"
lappend type_of_send "COASTAL"
lappend type_of_send "RIVER"
lappend type_of_send "TIDAL"
lappend type_of_send "OFFSHORE"
if {$argc != 1} {
  set errs  -1
  lappend errmsg "Usage ERROR:\nInvalid argument passed in.\nValid arguments are: "
  foreach xx $type_of_send {
    lappend errmsg "\t$xx"
  }
}
if {$argc == 1} {
  # make sure arg is correct
  set errs [lsearch $type_of_send [lindex $argv 0]]
  if {$errs == -1} {
    lappend errmsg "Usage ERROR:\nInvalid argument passed in: [lindex $argv 0].\nValid arguments
are:

"
    foreach xx $type_of_send {
      lappend errmsg "\t$xx"
    }
  }
}
if {$errs == -1 } {
  foreach xx $errmsg {
    puts stdout $xx
  }
  puts stdout "\r\r*** Program complete at: [clock format [clock seconds]]"
  exit;
}
#**********************************************************

# Initialize
#
set cnt 0
set ftpPrompt "ftp>"
set kbHitEnter "\r"
# Bump the timeout factor up so the program has enough time to transfer the files
# none should take longer then 2 minutes
set timeout 120

#*****************************************************************
# Site specific vars
#*****************************************************************
# The directory on ldad that has the mp3 files
set localDir "/data/ldad/voice"
# The directory on the web server where the mp3 files will be stored
set remoteDir "wxradio/voice"
# File extension (could be .wav, mp3, etc...)
set fileExt ".mp3"
# Servers (multiple servers Ok)
#   ER Server
lappend server "wwwftp.werh.noaa.gov"
# Username and password
#
#   The username and password have to line up with the server
#   EX] server[1] uses userPrompt[1],userName[1]and password[1]
#
lappend userPrompt "Name*:"
lappend userName "YOUR USER NAME\n"
lappend password "YOUR PASSWORD\n"

##############################
# Build the list of products #
##############################
if {[lindex $argv 0] == [lindex $type_of_send 0]} {
  lappend products "PHLHWRNW1"
  lappend products "PHLSAFNW1"
  lappend products "PHLHWRNW2"
  lappend products "PHLSAFNW2"
  lappend products "PHLHWRNW3"
  lappend products "PHLSAFNW3"
  lappend products "PHLHWRNW4"
  lappend products "PHLSAFNW4"
  lappend products "PHLHWRNW5"
  lappend products "PHLSAFNW5"
  lappend products "PHLHWRNW6"
  lappend products "PHLSAFNW6"
  lappend products "PHLHWRNW7"
  lappend products "PHLSAFNW7"
}
if {[lindex $argv 0] == [lindex $type_of_send 1]} {
  lappend products "PHICLINW1"
  lappend products "PHICLINW2"
  lappend products "PHICLINW3"
  lappend products "PHICLINW4"
}
if {[lindex $argv 0] == [lindex $type_of_send 2]} {
  lappend products "PHLCWFNW1"
  lappend products "PHLCWFNW2"
  lappend products "PHLCWFNW3"
  lappend products "PHLCWFNW4"
  lappend products "PHLCWFNW5"
  lappend products "PHLCWFNW6"
  lappend products "PHLCWFNW7"
}
if {[lindex $argv 0] == [lindex $type_of_send 3]} {
  lappend products "PHLRVSNW1"
  lappend products "PHLRVSNW2"
  lappend products "PHLRVSNW5"
  lappend products "PHLRVSNW7"
}
if {[lindex $argv 0] == [lindex $type_of_send 4]} {
  lappend products "PHLTIDNW1"
  lappend products "PHLTIDNW2"
  lappend products "PHLTIDNW3"
  lappend products "PHLTIDNW4"
  lappend products "PHLTIDNW5"
  lappend products "PHLTIDNW6"
  lappend products "PHLTIDNW7"
}
if {[lindex $argv 0] == [lindex $type_of_send 5]} {
  lappend products "PHLOFFNW2"
  lappend products "PHLOFFNW3"
  lappend products "PHLOFFNW4"
  lappend products "PHLOFFNW5"
}
set number_of_products 0
if {[info exists products]} {
  set number_of_products [llength $products]
}
if { $number_of_products == 0} {
  puts stdout "\rNo products to transmit."
  puts stdout "\r\r*** Program complete at: [clock format [clock seconds]]"
  exit
}
####################################################
# Run a loop for each server
#
foreach loc $server {
  # Start ftp and log on to the web server
  spawn ftp $loc
  expect [lindex $userPrompt $cnt] {
    send [lindex $userName $cnt]
    expect "Password:" {
      send [lindex $password $cnt ]
      expect "230*$ftpPrompt" {
        # change to the web server directory where the mp3 files will be stored
        send "cd $remoteDir $kbHitEnter"
        expect "250*$ftpPrompt" {
          send "lcd $localDir $kbHitEnter"
          expect "Local*$ftpPrompt" {
            send "binary $kbHitEnter"
            expect "200*$ftpPrompt" {
              send "prompt off $kbHitEnter"
              expect "Inter*$ftpPrompt" {
                # Send the files
                foreach pp $products {
                  send "delete $pp$fileExt $kbHitEnter "
                  expect "$ftpPrompt"
                  send "put $pp$fileExt $kbHitEnter "
                  expect "200*$ftpPrompt"
                }
              }
            }
          }
        }
        # Log off
        send "bye $kbHitEnter"
      }
    }
  }
  incr cnt
}

# Finalize
set timeout 10
puts stdout "\n\n*** Program complete at: [clock format [clock seconds]]\n"
```

Appendix B: – index.html

This file listing is an example html file that can be used as a template.

```
<!doctype html public "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>
   National Weather Service - NOAA Weather All Hazards Radio - Hear it on the web
</title>
<link href="css/wxradioweb.css" rel="STYLESHEET" type="text/css" />
<!-- CONFIGURE THE MP3 FILES TO BE PLAYED IN THE SCRIPTS/GETMENUITEMS.JS FILE -->
<script src="scripts/getMenuItems.js" language="JavaScript" type="text/javascript"></script>
<script src="scripts/wxradioweb.js"   language="JavaScript" type="text/javascript"></script>
<script src="scripts/mp3popup.js"     language="JavaScript" type="text/javascript"></script>
<noscript>
<h4 align="center">This web site will not work properly with your browser because Javascript is
not enabled.</h1>
</noscript>
</head>

<body>

<!--
    PUT LOCAL OFFICE INFORMATION IN HERE (this is optional)
    THE TOP AND LEFT ATTRIBUTES OF THE CLASS DEFINITION FOR nwrnote1
    (from css/wxradioweb.css) CONTROLS THE POSITION OF THIS BOX
    You can override the default settings here
-->
<div class="nwrnote1" style="top:50px; left:220px; height:70px;">
  <div style="font-weight:bold;text-decoration:underline;">
  <a href="http://www.erh.noaa.gov/ctp/wxradio/nwr.php" onClick="this.blur;">
  <span style="text-decoration:underline;color:black;">
  NOAA Weather Radio All Hazards</span></a>
  </div>
  Click on transmitter to select the<br /> product that you would like to hear
  <br />
  <!--
      CHANGE THE CONTACT EMAIL ADDRESS HERE
  -->
  <a class="cat" href="mailto:PUT_YOUR_EMAIL_ADDRESS_HERE@noaa.gov?subject=Question to the
National Weather -- NOAA Weather Radio All Hazards -- On the Web" style="font-
size:8pt;color:navy;" onClick="this.blur();">Question?</a>
</div>

<!--
     PUT LOCAL OFFICE MAP HERE
-->
<img src="images/nwrwebmap.png" usemap="#nwrmap"  id="img1" alt="NWR Coverage Map" width="216"
height="144" border="0" />

</body>
</html>

<!--
     PUT LOCAL OFFICE IMAGE MAP HERE
-->
<map id="nwrmap" name="nwrmap">
<area onClick="showMenu(this, event, miNW1, '250px')" onMouseOver="changeCursor('pointer')"
onMouseOut="changeCursor('auto')" alt="Tranmitter 1" shape="circle" coords="32,26,14" />
<area onClick="showMenu(this, event, miNW2, '250px')" onMouseOver="changeCursor('pointer')"
onMouseOut="changeCursor('auto')" alt="Tranmitter 2" shape="circle" coords="107,117,13" />
</map>
```

Appendix C: – css/wxradioweb.css

The following Cascading Style Sheet definitions are required. They control the appearance and screen placement of several elements used by this application.

```
A:link {
  color: #0000FF;
  text-decoration: none;
}
A:Visited  {
  color: #003399;
  text-decoration: none;
}
#wxRadioMenuBox{
  position:absolute;
  border:1px solid black;
  font-weight:normal;
  font-size: 9pt;
  font-family: Verdana, arial, courier;
  color: navy;
  line-height:18px;
  z-index:100;
}
#wxRadioMenuBox A{
  display: block;
  text-indent: 3px;
  color:navy;
}
#wxRadioMenuBox a:hover{ /*hover background color*/
  background-color: yellow;
}
.nwrNote1 {
  font-size: 8pt;
  font-family: arial, courier, sans-serif;
  position: absolute;
  top : 50px;
  left: 500px;
  background-color: white;
  width: 250px;
  height: 48px;
  border: 2px double black;
  text-align:center;
}
.cat {
  font-size: 10pt;
  font-family: "Trebuchet MS", Arial,Helvetica, San Serif;
  color: #0000cd;
  text-decoration: none;
}

A.cat:link {
  font-size: 10pt;
  font-family: "Trebuchet MS", Arial,Helvetica, San Serif;
  color: #0000cd;
  text-decoration: none;
}
A.cat:visited {
  font-size: 10pt;
  font-family: "Trebuchet MS", Arial,Helvetica, San Serif;
  color: #0000cd;
  text-decoration: none;
}
A.cat:hover {
  font-size: 10pt;
  font-family: "Trebuchet MS", Arial,Helvetica, San Serif;
  color: #ff0000;
  background-color: #ffff99;
}
A.cat:link {
  font-size: 10pt;
  font-family: "Trebuchet MS", Arial,Helvetica, San Serif;
  color: #0000cd;
  text-decoration: none;
}
```

Appendix D: – scripts/getMenuItems.js

This file defines the drop down menus used by this application. Areas that need to be modified by the WFO are highlighted. The transmitter arrays determine the products that can be selected by each transmitter. The array name is referenced by the image map call to showMenu().

```
//
// THE LOCATION OF THE MP3 FILES GOES HERE
// IT CAN BE A FULLY QUALIFIED LOCATION OR A REFERENCE TO A RELATIVE LOCATION
//  i.e, FULLY QUALIFIED: voiceFileLocation = 'http://weather.gov/er/phi/wxradio/voice/';
//              RELATIVE: voiceFileLocation = 'voice/';
voiceFileLocation = 'voice/';
linkPart1 = '<a href="' + voiceFileLocation;
playerFunction = "PlayerOpen('";
linkPart2 = '" target="_blank" onclick="javascript:' + playerFunction;
linkPart3 = "' ,this.href); return false" + '">';


// Transmitter NW1
var miNW1 = new Array();
miNW1.push( linkPart1 + 'PHLHWRNW1.mp3' + linkPart2 +'Hourly Weather Roundup<br>Allentown, PA'+
linkPart3 + 'Hourly Weather Roundup</a>');
miNW1.push( linkPart1 + 'PHLSAFNW1.mp3' + linkPart2 +'Local Forecast<br>Allentown, PA'+ linkPart3
+ 'Local Forecast</a>');
miNW1.push( linkPart1 + 'PHICLINW1.mp3' + linkPart2 +'Climate Summary<br>Allentown, PA'+
linkPart3 + 'Climate Summary</a>');
miNW1.push( linkPart1 + 'PHLRVSNW1.mp3' + linkPart2 +'River Stage<br>Allentown, PA'+ linkPart3 +
'River Stage</a>');
miNW1.push( linkPart1 + 'PHLCWFNW1.mp3' + linkPart2 +'Coastal Waters Forecast<br>Allentown, PA'+
linkPart3 + 'Coastal Waters Forecast</a>');
miNW1.push( linkPart1 + 'PHLTIDNW1.mp3' + linkPart2 +'Tidal Information<br>Allentown, PA'+
linkPart3 + 'Tidal Information</a>');
// Transmitter NW2
var miNW2 = new Array()
miNW2.push( linkPart1 + 'PHLHWRNW2.mp3' + linkPart2 +'Hourly Weather Roundup<br>Philadelphia,
PA'+ linkPart3 + 'Hourly Weather Roundup</a>');
miNW2.push( linkPart1 + 'PHLSAFNW2.mp3' + linkPart2 +'Local Forecast<br>Philadelphia, PA'+
linkPart3 + 'Local Forecast</a>');
miNW2.push( linkPart1 + 'PHICLINW2.mp3' + linkPart2 +'Climate Summary<br>Philadelphia, PA'+
linkPart3 + 'Climate Summary</a>');
miNW2.push( linkPart1 + 'PHLRVSNW2.mp3' + linkPart2 +'River Stage<br>Philadelphia, PA'+ linkPart3
+ 'River Stage</a>');
miNW2.push( linkPart1 + 'PHLCWFNW1.mp3' + linkPart2 +'Coastal Waters Forecast<br>Philadelphia,
PA'+ linkPart3 + 'Coastal Waters Forecast</a>');
miNW2.push( linkPart1 + 'PHLOFFNW2.mp3' + linkPart2 +'Offshore Waters Forecast<br>Philadelphia,
PA'+ linkPart3 + 'Offshore Waters Forecast</a>');
  miNW2.push( linkPart1 + 'PHLTIDNW1.mp3' + linkPart2 +'Tidal Information<br>Philadelphia, PA'+
                              linkPart3 + 'Tidal Information</a>');
```

Appendix E: – scripts/wxradioweb.js

This file controls the behavior of the drop down menus utilized in the nwrweb.html page. Areas that must be changed by the WFOs is highlighted.

```
// Default values
var MenuWidth='200px'          // default menu width
var menubgcolor='lightyellow'  // menu bgcolor
var HideMenuDelayFactor=250    // menu disappear speed onMouseout (in miliseconds)
var hidemenu_onclick="yes"     // hide menu when user clicks within menu?
var CurrentMenu = new Object();
//
// Check the browser to ensure that required feature will work
var ie4 = document.all
var w3DOM = document.getElementById
//
// Redirect to a text only page if required options are not supported
//
if (ie4||w3DOM) {
  document.write('<div id="wxRadioMenuBox"
style="visibility:hidden;width:'+MenuWidth+';background-color:'+menubgcolor+'"
onMouseover="ResetTimerValue()" onMouseout="dynamichide(event)"></div>');
}
else {
  window.location.href = 'PUT ALTERNATE PAGE HERE';
}


function showMenu(obj, e, MenuOptions, MenuWidth){
  // stop the event from propagating
  if (window.event)
    event.cancelBubble=true;
  else
    if (e.stopPropagation)
      e.stopPropagation();
  //
  ResetTimerValue() ;
  //
  CurrentMenu=document.getElementById? document.getElementById("wxRadioMenuBox") : wxRadioMenuBox
  LoadMenu(MenuOptions)
  ShowOrHideMenu(CurrentMenu.style, e, "visible", "hidden", MenuWidth)
  //
  // Place the menu
  //
  // 1. Get the position that the mouse is when this event fired
  var MenuLeftPosition = getMouseX(e);
  var MenuTopPosition  = getMouseY(e);
  // 2. Determine if the menu should be displayed left/right/up/down
  //MenuLeftPosition = AdjustForWidth(MenuLeftPosition, MenuWidth);
  //MenuTopPosition = AdjustForHeight(MenuLeftPosition);
  CurrentMenu.style.left = MenuLeftPosition + "px"
  CurrentMenu.style.top  = MenuTopPosition + "px"
return false;
}

function AdjustForWidth(startLeft, menuwidth) {
  //startLeft is where the mouseOver occurred
  //menuwidth is the width of the menu object
  var borderFactor = 20;
  vStart = startLeft;
  var WindowWidth = document.body.offsetWidth ? document.body.offsetWidth : 0;
  if (WindowWidth > 0) {
    if ((vStart + menuwidth + borderFactor) > WindowWidth)
      vStart = vStart - menuwidth;
  }
  var ans = vStart;
  return ans;
}

function AdjustForHeight(varIn) {
  var height = document.body.offsetHeight ? document.body.offsetHeight : 0;
  var ans = varIn;
  return ans;
}

function ShowOrHideMenu(obj, e, visible, hidden, MenuWidth){
  if (MenuWidth!=""){
    CurrentMenu.widthobj=CurrentMenu.style;
    CurrentMenu.widthobj.width=MenuWidth;
  }
  if (e.type=="click" && obj.visibility==hidden || e.type=="mouseover")
    obj.visibility=visible;
  else
    if (e.type=="click")
      obj.visibility=hidden;
  return true;
}

function LoadMenu(what){
  // Make the menu elements one string
  if (ie4)
    CurrentMenu.innerHTML=what.join(" ");
  else
    document.getElementById("wxRadioMenuBox").innerHTML=what.join(" ");
}

function contains_ns6(a, b) {
  while (b.parentNode)
    if ((b = b.parentNode) == a)
      return true;
  return false;
}

function dynamichide(e){
  if (ie4&&!CurrentMenu.contains(e.toElement))
    delayhidemenu()
  else
    if (w3DOM&&e.currentTarget!= e.relatedTarget&& !contains_ns6(e.currentTarget,
e.relatedTarget))
      delayhidemenu()
}

function hidemenu(e){
  if (typeof CurrentMenu!="undefined")
    CurrentMenu.style.visibility="hidden"
}

function delayhidemenu(){
  delayhide=setTimeout("hidemenu()",HideMenuDelayFactor)
}

function ResetTimerValue(){
  if (typeof delayhide!="undefined")
    clearTimeout(delayhide)
}

function getMouseX(e){
var posx;
  posx=0;
  if(!e) var e=window.event;
  if(e.pageX||e.pageY){
    posx=e.pageX;
  }
  else
    if(e.clientX||e.clientY){
      posx = e.clientX +  + getScrollX();
    }
  return posx;
}

function getMouseY(e){
var posy;
  posy=0;
  if(!e) var e=window.event;
  if(e.pageY){
    posy=e.pageY;
  }
  else
    if(e.clientY){
      posy = e.clientY  + getScrollY();
    }
  return posy;
}

function getScrollX() {
  var scrOfX = 0, scrOfY = 0;
  if( typeof( window.pageYOffset ) == 'number' ) {
    //Netscape compliant
    scrOfY = window.pageYOffset;
    scrOfX = window.pageXOffset;
  } else if( document.body && ( document.body.scrollLeft || document.body.scrollTop ) ) {
    //DOM compliant
    scrOfY = document.body.scrollTop;
    scrOfX = document.body.scrollLeft;
  } else if( document.documentElement &&
      ( document.documentElement.scrollLeft || document.documentElement.scrollTop ) ) {
    //IE6 standards compliant mode
    scrOfY = document.documentElement.scrollTop;
    scrOfX = document.documentElement.scrollLeft;
  }
  return scrOfX;
}

function getScrollY() {
  var scrOfX = 0, scrOfY = 0;
  if( typeof( window.pageYOffset ) == 'number' ) {
    //Netscape compliant
    scrOfY = window.pageYOffset;
    scrOfX = window.pageXOffset;
  } else if( document.body && ( document.body.scrollLeft || document.body.scrollTop ) ) {
    //DOM compliant
    scrOfY = document.body.scrollTop;
    scrOfX = document.body.scrollLeft;
  } else if( document.documentElement &&
      ( document.documentElement.scrollLeft || document.documentElement.scrollTop ) ) {
    //IE6 standards compliant mode
    scrOfY = document.documentElement.scrollTop;
    scrOfX = document.documentElement.scrollLeft;
  }
  return scrOfY ;
}

function changeCursor( valIn ) {
  // possible values auto,pointer,crosshair,help,move,text,default,wait
  // possible values auto,pointer,crosshair,help,move,text,default,wait
  if (ie4)
    document.body.style.cursor = valIn;
  else if (w3DOM)
    document.getElementById('img1').style.cursor = valIn;
}

if (hidemenu_onclick=="yes")
  document.onclick=hidemenu
```

Appendix F: – scripts/mp3popup.js

This file controls how the browser embedded mp3 player should behave.

```
// mp3popup.js
var UniqueID = 314 // Make each link open in a new window.
var newWinOffset = 0 // Position of first pop-up

function PlayerOpen(soundfiledesc,soundfilepath) {
  PlayWin = window.open('',UniqueID,'width=320,height=190,top=' + newWinOffset
+',left=0,resizable=0,scrollbars=0,titlebar=0,toolbar=0,menubar=0,status=0,directories=0,personal
bar=0');
  PlayWin.focus();
  var winContent = "<HTML><HEAD><TITLE>" + soundfiledesc + "</TITLE></HEAD><BODY
bgcolor=#ffffcc>";
  winContent += "<B style='font-size:18px;font-family:Verdana,sans-serif;line-height:1.5'>" +
soundfiledesc + "</B>";

  winContent += "<OBJECT width='300' height='42'>";
  winContent += "<param name='SRC' value='" +  soundfilepath + "'>";
  winContent += "<param name='AUTOPLAY' VALUE='true'>";
  winContent += "<param name='CONTROLLER' VALUE='true'>";
  winContent += "<param name='BGCOLOR' VALUE='#FFFFCC'>";
  winContent += "<EMBED SRC='" + soundfilepath + "' AUTOSTART='TRUE' LOOP='FALSE' WIDTH='300'
HEIGHT='42' CONTROLLER='TRUE' BGCOLOR='#FFFFCC'></EMBED>";
  winContent += "</OBJECT>";
  winContent += "<FORM><DIV align='center'><INPUT type='button' value='Close this window'
onClick='javascript:window.close();'></DIV></FORM>";

  winContent += '<div style="font-size:8pt;color:#2f4f4f;text-align:center;"><a
href="nwrwebhelp.html" class="cat">Help</a></div>';

  winContent += "</BODY></HTML>";
  PlayWin.document.write(winContent);
  PlayWin.document.close(); // "Finalizes" new window
  UniqueID = UniqueID + 1
  // newWinOffset = newWinOffset + 20 // subsequent pop-ups will be this many pixels lower.
}
```