xapmcsImage

Digital Frame Image & RSS Feed

Collect and Merge Image Data with Text Overlays

Michael McSharry
Jan 15, 2010

Table of Contents

Table of Figures

Table of Tables

None

1 Introduction

xapmcsImage is intended to produce custom images for display on digital photo frames. It includes an edit function to identify the contents of constructed images, data gathering functions, text overlay, expression evaluation, periodic reconstruction and service of constructed RSS feeds.

The application runs standalone and will accept xAP message data as a source of information used for text and icon overlays. The text overlay can be formed with literal strings and as the result of expressions from received data.

Images are obtained from the file system, internet downloads and as a result of chart requests via xapmcsChart.

Image processing capabilities consists of positioning, sizing and merging. Text processing allows specification of color, transparency, font, size and style.

xapmcsImage it is a standalone application that can be used to statically build images and RSS xml files. For effective use in generating images based upon dynamic data an xAP environment is needed. For the Homeseer user this will consist of mcsXap Homeseer plugin that will make available Homeseer information via xAP protocol and an xAP hub that will route xAP messages within a PC.

2 xapmcsImage Setup

2.1 Top Level Organization

The contents of the zip package should be unzipped into a convenient location such as C:\Program Files\xAP\xapmcsImage\. The executable program is xapmcsImage.exe. It is a .NET application and provides no ActiveX/COM interface.

The install will deposit a cascading style sheet in the \HTML subfolder. This can be edited, if desired, to achieve a browser display that fits better with other applications. At the same location there will also be an .htm file that contains clickable links that will be shown at the top of the browser display to provide quick access to other applications. This will need to be edited to reflect any applications that are appropriate for your environment.

xapmcsImage is started by running xapmcsImage.exe. It will present itself as a tray icon such as is shown in Figure 1. Clicking on the Icon will bring up a menu from which various browser pages can be selected.

Figure 1 xapmcsImage Tray Icon

The tray icon, when left-clicked will make available four options. The first, xAP Data, is the page where setup options are selected and xAP inputs are selected. The second, RSS Image, is the editor where RSS feeds and Image canvases are specified. The others are support pages to view log messages and this manual as a help file.

2.2 Image Construction

Images are constructed from a set of sequential activities. The activities are specified from the RSS Image browser selection.

Each activity has an associated set of parameters. The parameters can be literal numbers and text or they can be based upon dynamic data received on the LAN. Expressions can be used for any parameter that resolves to a number. For example, the X coordinate of an item can be specified as “100” or as “60+40”. It can also be specified as a function of dynamic data such as “{StartingPoint}+20”.

Any parameter that resolves to text will be entered as a pull-down selection with three exceptions. One is the color and transparency which can be either a literal hex value or based upon dynamic data. For example “80FF9070” or “{BackgroundColor}” are acceptable entries for a color. The second is the Filename or URL used where again the literal value or a dynamic value can be entered. The third is the expression used in the TEXT activity. In this case a combination of numeric expressions, dynamic data, and string concatenation can be used.

Initially a table of ten rows is presented. The first column of each row identifies the type of activity and the remaining columns are parameters associated with the activity. The activities are performed in the row order of the table.

The activities supported are RSS, CANVAS, GRAPH, URL, CROP, NOCROP, IMAGE, BOX, FONT, and TEXT.

RSS allows a user to specify an RSS server other than xapmcsImage. It is also the place where the rate of reconstruction is identified. The reconstruction consists of all images and the RSS xml file. The RSS feed and supporting images will not be built if the reconstruction rate is left blank. This feature can be used to temporarily stop reconstruction of a specific RSS feed. The rate is specified as the number of seconds. The RSS server supported by xapmcsImage is the same port as the browser and this port is specified on the xAP Data browser page or manually edited in \Config\xapmcsImage.ini should the default port 8026 not be initially reachable.

CANVAS allows a user to identify the size of the constructed image. The default is 800x480 which is the size for the Kodak WiFi W820 or W1020 frame. It also provides an second place where the reconstruction rate is identified. This is only necessary if an RSS activity is not specified. The last parameter is the output filename. This can be any location on the LAN using UNC convention. If left blank the image will be constructed in the \HTML folder under the xapmcsImage install.

GRAPH is used to make a request to xapmcsChart to construct a chart image. The parameters consist of those elements needed by xapmcsChart to identify the chart to be constructed. A wait parameter is used to specify a timeout period. xapmcsImage will look at 250 millisecond intervals for the chart to be constructed and will give up at the Wait seconds inputs. The default wait is 10 seconds and is invoked if the Wait field is blank. The chart output file can be specified or if left blank the output will be placed in the \HTML subfolder.

URL is used to download an image from the internet or other local server and also can be used to retrieve an image from the file system. The URL parameter is the full http (or ftp) Universal Resource Locator for the image to be downloaded. A Wait parameter can also be used to limit the wait for the image to download. If not specified a wait of 10 seconds is enforced. For local file system a UNC or local file path convention can be used.

CROP is used as a precursor to the IMAGE tag to define the subset of the full image generated by the URL, GRAPH or IMAGE activities that will be placed on the canvas. The CROP is used to select a subset of a source image or remove unwanted segments of a source image. Cascading CROP activities are not supported directly, but can be achieved by using the output of one CANVAS as the input to a second where further cropping can be done.

NOCROP is used following the IMAGE command to remove the copping window of subsequent source images. It is only needed if multiple IMAGE commands are used on the same canvas. NOCROP is the default which means the source image is resized in accordance with the IMAGE activity.

IMAGE is used to position an image on the canvas and optionally alter the size. The location is specified with starting X (horizontal) and Y (vertical) coordinate. The canvas coordinate system is (X=0, Y=0) at upper left and (X=Width, Y=Height) as lower left where Width and Height are identified with the CANVAS line. Note that no negative coordinates are available.

If the IMAGE processing is associated with a GRAPH or URL image then the File path parameter will normally be left blank. If it associated with a file system file then this filename will be entered. Note that either the URL or IMAGE activities can be used to obtain an existing image from the LAN.

BOX is used to define a rectangular element that is normally used to constrain the overlay of text on the image. Its parameters consist of the location (X,Y) and size (W,H) as well as the background color within the box. The color is specified as an alpha channel for transparency and RGB for color. This is an 8 character hex as AARRGGBB. An AA of 00 is used for total transparency. The default box is full transparency with same size as canvas.

FONT is used to define the characteristics of text overlay. The parameters are all pull-down selections with the exception of the font size which is numeric point representation. Default font is TBD

TEXT is used to overlay text on top of other images on the canvas. The text parameters specify positioning within the most recently defined BOX rectangle and an expression that will resolve to a text string. The result of the expression will be the overlaid text.

2.3 Parameter Expressions

A fixed text overlay is entered using leading and training regular quote (“). For example, “ABC 123”. The ampersand character can be used to concatenate text segments. For example “ABC” & “-“ & “123”.

Numeric components of the expression use VB syntax. For example, Round(1/3,2) would be used to divide 1 by 3 and show the result with 2 significant digits (i.e. 0.33). Numeric and literal expressions can be combined. For example “AB”&Round(1/3,2)&”Degrees F”

Conditional expressions have the form “If condition | true condition substitution | false condition substitution”. The first three characters are “If “. The “|” is used to separate the three elements of the IF/THEN/ELSE expression. Note that alias substitution (described below) can be used in any part of the IF/THEN/ELSE expression.

The elements of expressions can also come from data received in xAP messages. The xAP data page is where the data of interest is identified. Any data value that will be used as part of a TEXT expression will be identified with an Alias name. This name will then be used in the expression. “{}” are used to encase the alias to distinguish it from other parts of the expression. Assume an alias has been defined called Occupancy and the data expected will be either “Home” or “Away”. The following expression will produce a text string that will show the current status and the time it changed to this status.

“House Status is ”&{Occupancy}&” since “&{Occupancy.Date}

Note in the prior example the {Occupancy.Date} vs. {Occupancy}. When on {Occupancy} is used then the data as received in the xAP message will be used. When a “.xxx” property is include then a characteristic of this alias will be used. The available characteristics are String, Text, Value, Image, and Date.

“String” is full value of the received data. It is equivalent to {Alias}.
“Text” is the received data excluding everything contained within HTML tags.
“Value” is the same as “Text” with the further restriction that only the numeric portion is used. When used as part of computations then the “.Value” should always be used to assure valid data in the computation.
“Image” will identify the target of the <img src…> and expand the target path to include the path of the xapmcsImage HTML folder. It does not retrieve the img file, but it does provide a way to reference it in expressions. The file will need to be manually copied to

the HTML folder. As an example, assume that HS has ON.GIF in its status string and this ON.GIF file is located in the Homeseer\HTML folder. This ON.GIF file will need to be manually copied to the xapmcsImage HTML folder. When the RSS frame image is constructed, xapmcsImage will pull this ON.GIF file from its \HTML should the {Alias.Image} parameter be used as part of an expression. Note that {Alias.Image} is not used as a TEXT element, but only as the File parameter of the IMAGE or URL tags.

There are some common aliases and characteristics that are internally defined and not the result of a xAP message item. These are the {Now} alias and the ”FormatDate” and “FormatNumber” characteristics. Format uses the same formatting convention as the .NET ToString formatter. See http://msdn.microsoft.com/en-us/library/427bttx3(VS.71).aspx For example the following shows the day of week and time to the nearest minute for the current time {Now.Format(ddd HH:mm)}. The Format characteristic can be applied to any alias such as {Occupancy.Date.FormatDate(d)} or {Cost.FormatNumber(c)}

3 RSS Image Browser Page

The RSS Image browser page supports the construction of images as defined in Section Image Construction2.2.

In addition to the described fields a checkbox is available on each row. This checkbox is used to insert a new row above the checkboxed row. The new row will default to TEXT which can subsequently be changed to whatever activity is desired.

A row is removed by changing the activity to blank.

All edit changes are saved to the xapmcsImage.ini file when the Save Edits button is used. While this editing is underway there is no change to the periodic construction of images and RSS feeds that were previously started. The “Apply Changes” button is used to actually start producing new images and RSS files based upon the edits.

The construction of an image must start with a RSS or CANVAS activity and at least one of the two are required. The TEXT activity will follow earlier FONT and BOX activity unless the default properties are adequate. The most recent FONT and BOX is used until another is defined. A URL or GRAPH activity will be followed by an IMAGE activity to place the downloaded or constructed image on the canvas. The TEXT activity will follow other activities where text is to be overlaid. The last activity performed that places something on the canvas will supersede other activities that occupy the same pixels on the canvas.

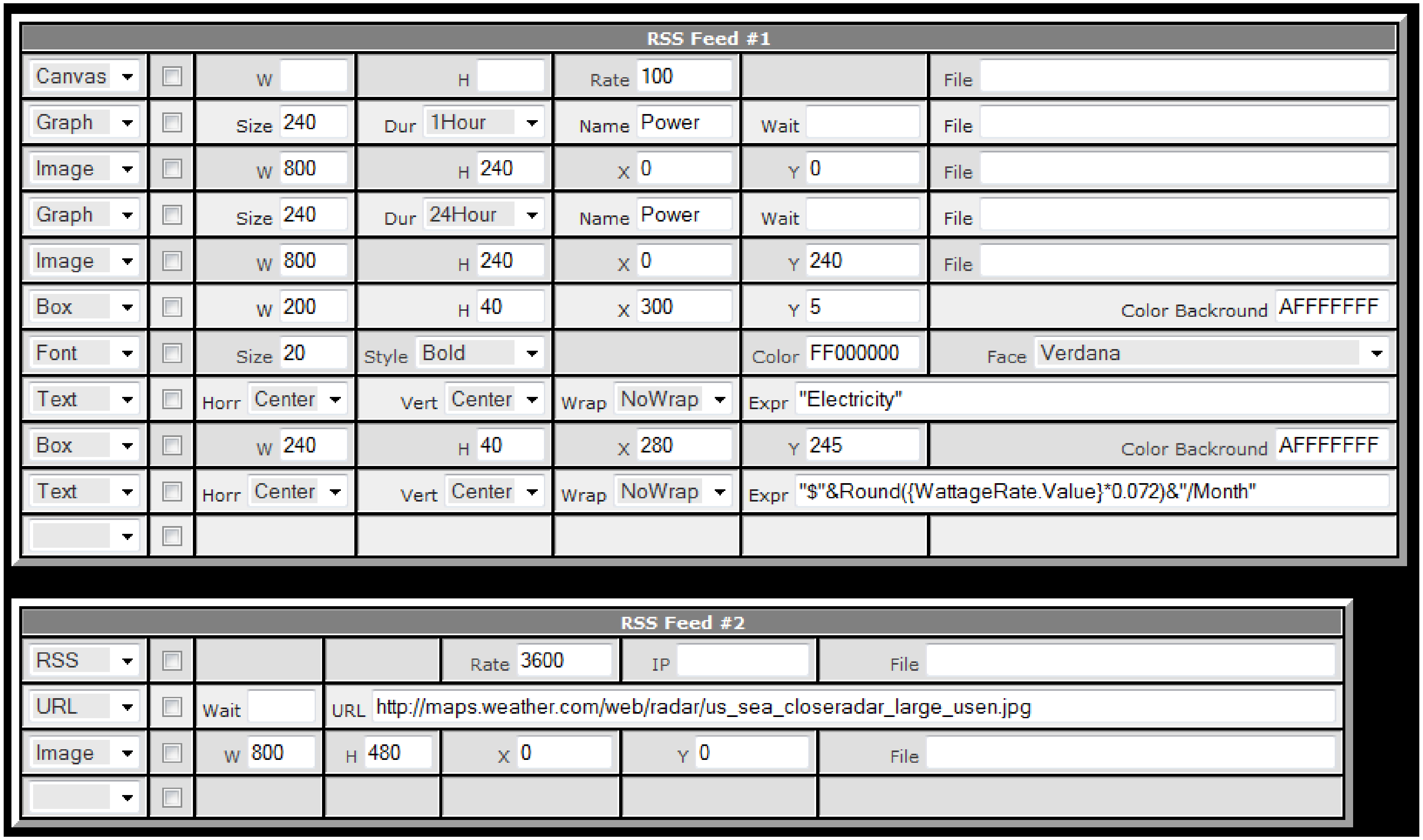
An RSS feed is constructed with a boilerplate header and one item for each CANVAS tag in the RSS feed specification. A Digital Frame will typically cycle through each of the items (canvas images) at the cycle rate setup for the frame. The Frame will typically retrieve the RSS feed after showing all items and after some delay implemented in the Frame firmware.

Figure 2 through Figure 6 provide an example a definition of two RSS feeds, the resultant image and xml. The first RSS feed was built by drawing two graphs and then placing two lines of text with a semi-opaque background. The second line of text is based upon a calculation performed using received xAP data. This calculation can be seen in the second TEXT tag. The first RSS feed is rebuilt every 100 seconds.

RSS Feed #1 starts with CANVAS activity where the default of 800x480 is used for W and H and the reconstruction rate of 100 seconds is entered. This is followed by two pairs of GRAPH/IMAGE activities. The graph is the "Power" group setup in xapmcsChart using the screen size of "240" also setup in xapmcsChart. The first graph is constructed based upon chart duration of 1 hour and the second is for 24 hours. The IMAGE activities resize these to 800x240 and place them as stacked images on the canvas.

The end of the RSS Feed #1 defines the text overlays. The FONT activity of 20 point, Bold, Horizontal, Opaque Black and Verdana face are used for both. The BOX is used to position the text and apply a background color to assure the text does not blend into the underlying image. A transparent background can also be used as is illustrated later with RSS Feed #3.

There are two TEXT activities. The first is simple literal "Electricity" that is centered in the top box. The second is a calculation that uses the instantaneous wattage rate obtained from xAP message on the LAN. This expression prepends a "$" and appends the "/Month" interval. The calculation multiplies the 0.072 by the numeric portion (i.e. .Value property) of the received data identified by the Alias {WattageRate}.



RSS Feed #1

Canvas	☐	W	H	Rate 100		File
Graph	☐	Size 240	Dur 1Hour	Name Power	Wait	File
Image	☐	W 800	H 240	X 0	Y 0	File
Graph	☐	Size 240	Dur 24Hour	Name Power	Wait	File
Image	☐	W 800	H 240	X 0	Y 240	File
Box	☐	W 200	H 40	X 300	Y 5	Color Backround AFFFFFFF
Font	☐	Size 20	Style Bold		Color FF000000	Face Verdana
Text	☐	Horr Center	Vert Center	Wrap NoWrap	Expr "Electricity"	
Box	☐	W 240	H 40	X 280	Y 245	Color Backround AFFFFFFF
Text	☐	Horr Center	Vert Center	Wrap NoWrap	Expr "$"&Round({WattageRate.Value}*0.072)&"/Month"	
	☐					

RSS Feed #2

RSS	☐			Rate 3600	IP	File
URL	☐	Wait	URL http://maps.weather.com/web/radar/us_sea_closeradar_large_usen.jpg			
Image	☐	W 800	H 480	X 0	Y 0	File
	☐					

Figure 2 RSS Image Page Example

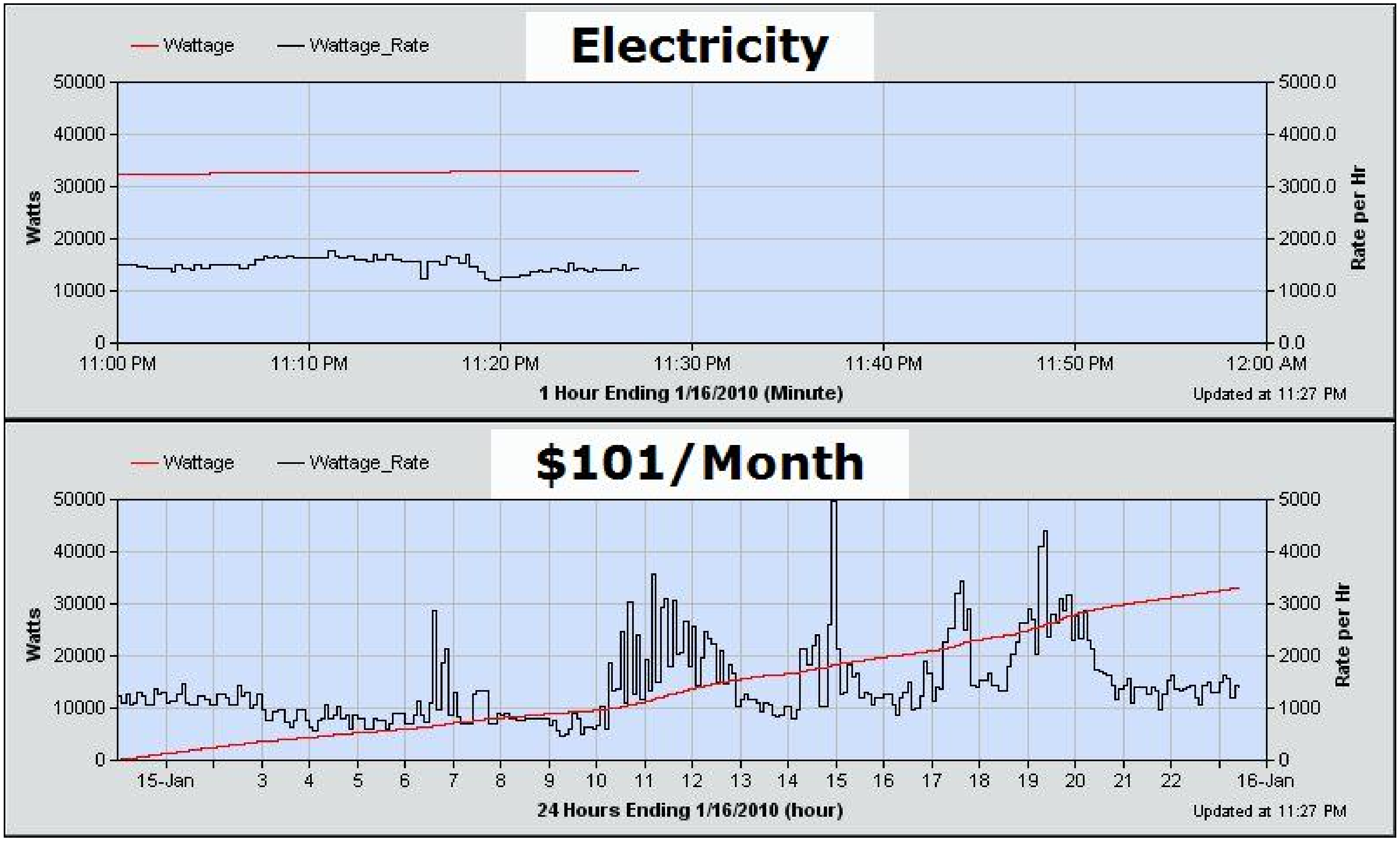
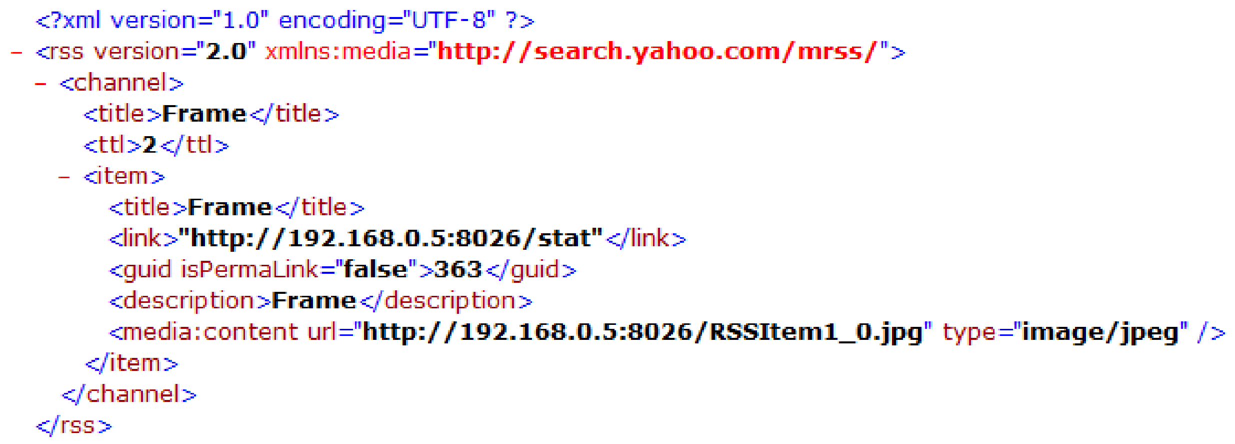


Figure 3 RSS #1 Results Example



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <rss version="2.0" xmlns:media="http://search.yahoo.com/mrss/">
  - <channel>
      <title>Frame</title>
      <ttl>2</ttl>
    - <item>
        <title>Frame</title>
        <link>"http://192.168.0.5:8026/stat"</link>
        <guid isPermaLink="false">363</guid>
        <description>Frame</description>
        <media:content url="http://192.168.0.5:8026/RSSItem1_0.jpg" type="image/jpeg" />
      </item>
    </channel>
  </rss>
```

Figure 4 RSS #1 XML Example

The second feed simply downloads a weather radar image and resizes it to fit the full frame. In this case the RSS activity was used to specify the update rate of every hour. The IP and File properties were left blank to indicate that xapmcsImage will be serving the RSS xml file and the default xml filename will be used.

The URL activity identifies where the internet image will be obtained. The default 10 second maximum wait will be used. The IMAGE activity then resizes the downloaded image to 800x480 and positions it at the upper left.

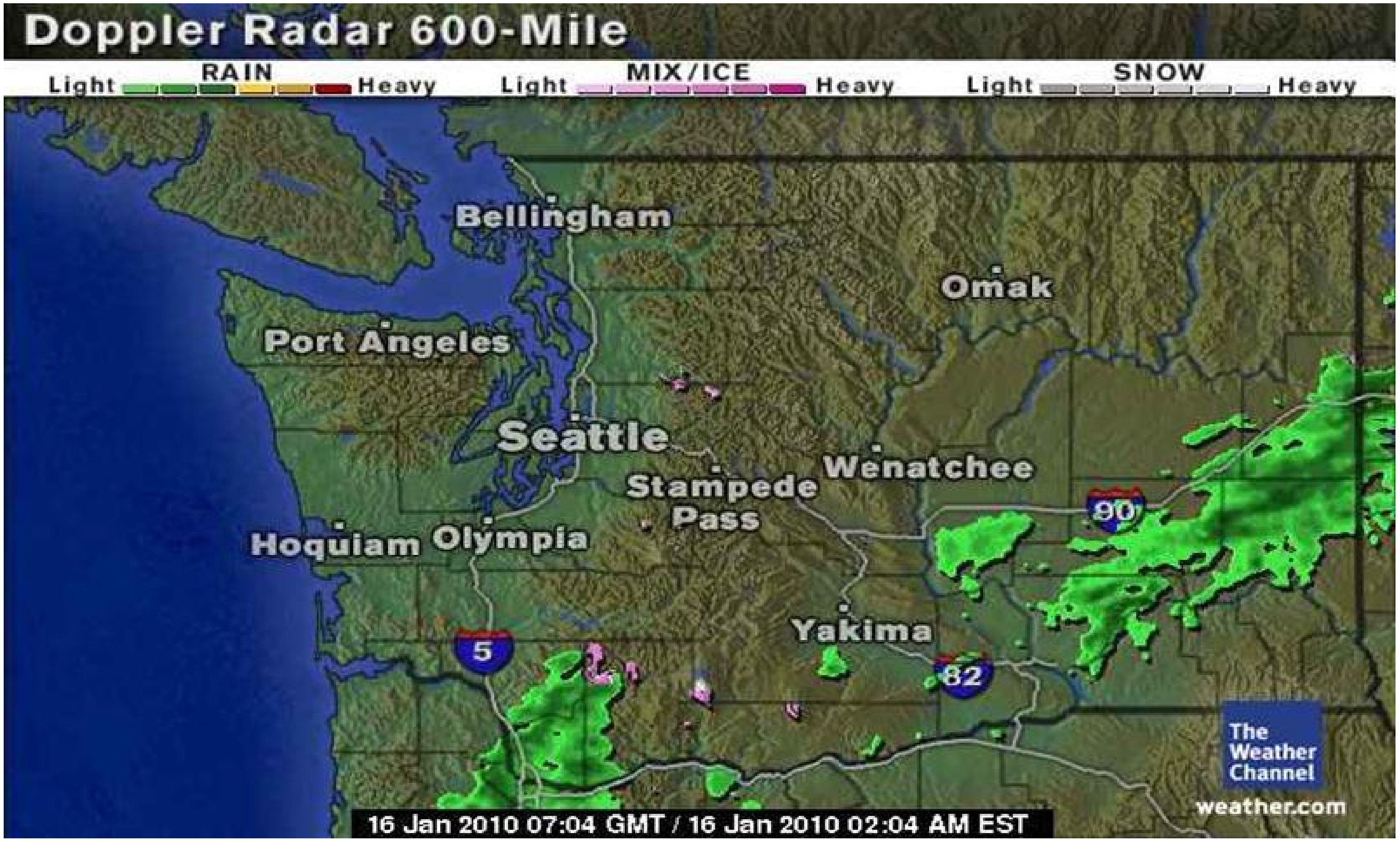
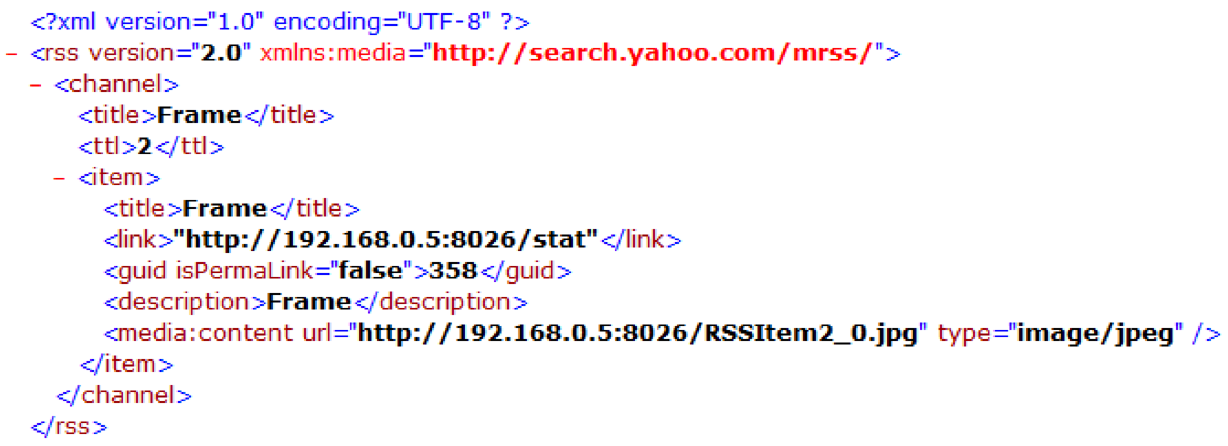


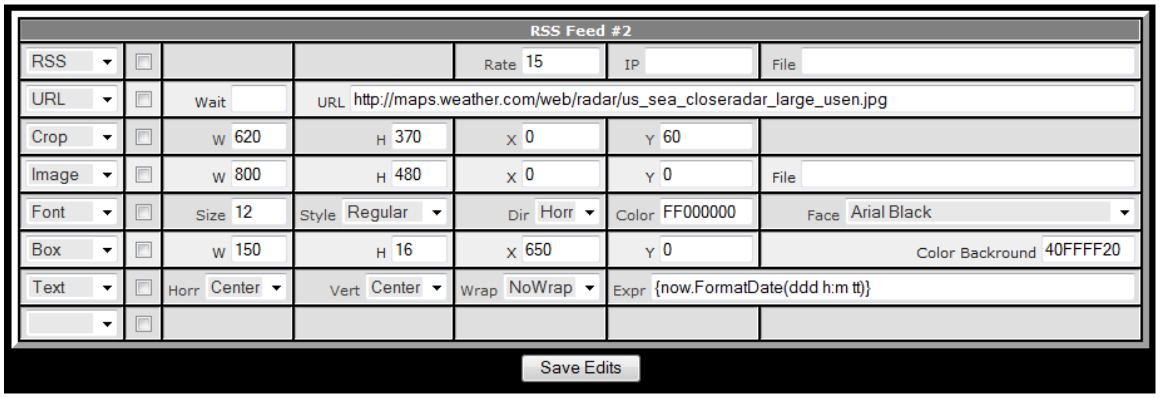
Figure 5 RSS #2 Results Example



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <rss version="2.0" xmlns:media="http://search.yahoo.com/mrss/">
  - <channel>
      <title>Frame</title>
      <ttl>2</ttl>
    - <item>
        <title>Frame</title>
        <link>"http://192.168.0.5:8026/stat"</link>
        <guid isPermaLink="false">358</guid>
        <description>Frame</description>
        <media:content url="http://192.168.0.5:8026/RSSItem2_0.jpg" type="image/jpeg" />
      </item>
    </channel>
  </rss>
```

Figure 6 RSS #2 XML Example

The downloaded radar image contains legend and other information that is not of interest for display and this can be removed with the CROP activity. The simple URL download RSS definition was augmented to crop off the undesired portion and to relabel the date stamp overlay to used local time in a more pleasing format. The definition and new image are shown in Figure 7 and Figure 8. The CROP activity takes a window of 620x370 starting at (0,60) from the image downloaded with the URL activity. The new lines of FONT, BOX and TEXT place black text on a semi-transparent yellow background in the upper left of the canvas. The TEXT expression uses the "Now" alias and the FormatDate characteristic to capture the current time and format it in the desired manner.



RSS Feed #2						
RSS	☐			Rate 15	IP	File
URL	☐	Wait	URL http://maps.weather.com/web/radar/us_sea_closeradar_large_usen.jpg			
Crop	☐	W 620	H 370	X 0	Y 60	
Image	☐	W 800	H 480	X 0	Y 0	File
Font	☐	Size 12	Style Regular	Dir Horr	Color FF000000	Face Arial Black
Box	☐	W 150	H 16	X 650	Y 0	Color Backround 40FFFF20
Text	☐	Horr Center	Vert Center	Wrap NoWrap	Expr {now.FormatDate(ddd h:m tt)}	
	☐					

Figure 7 Cropped and Timestamped Radar Image Definition

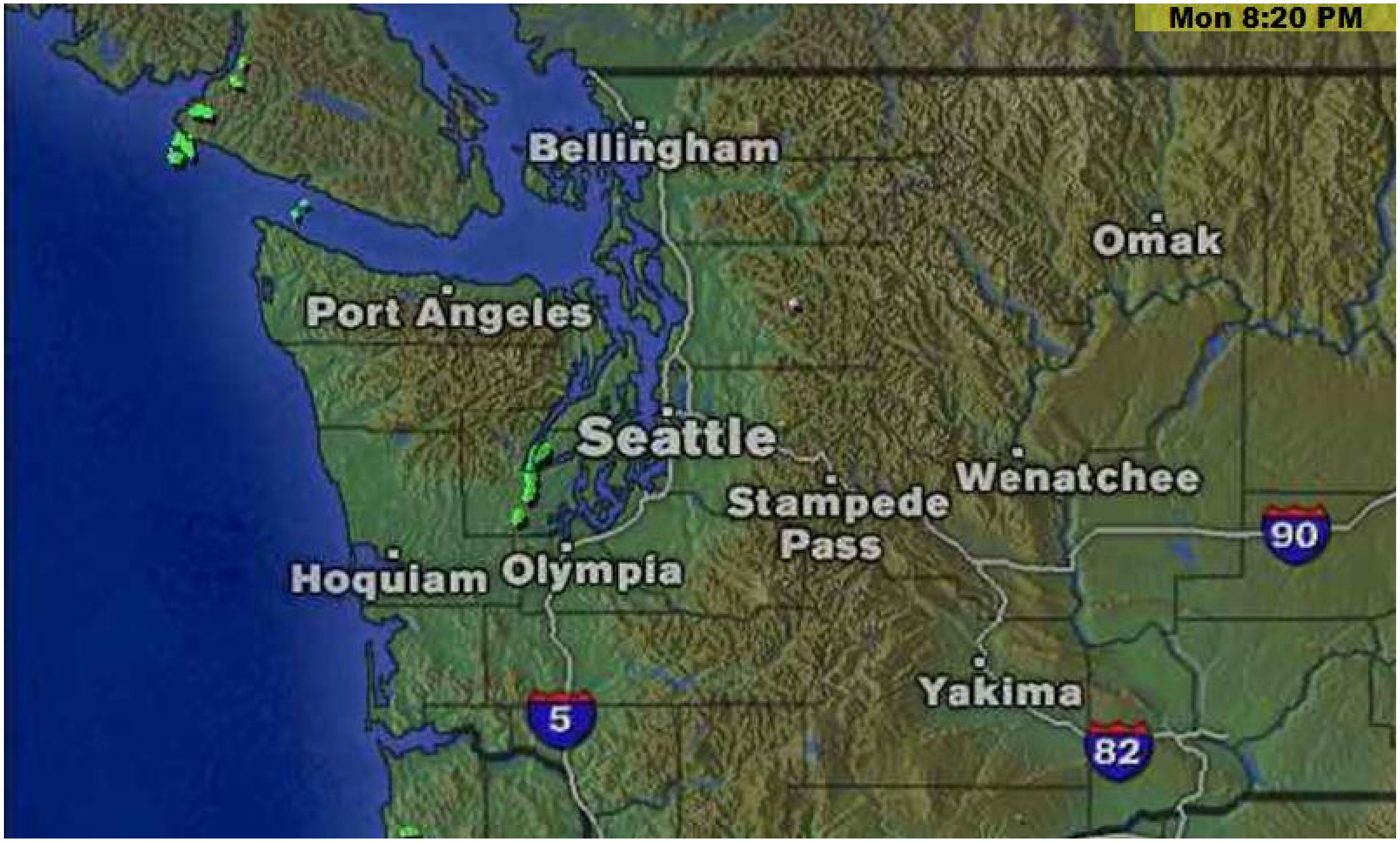


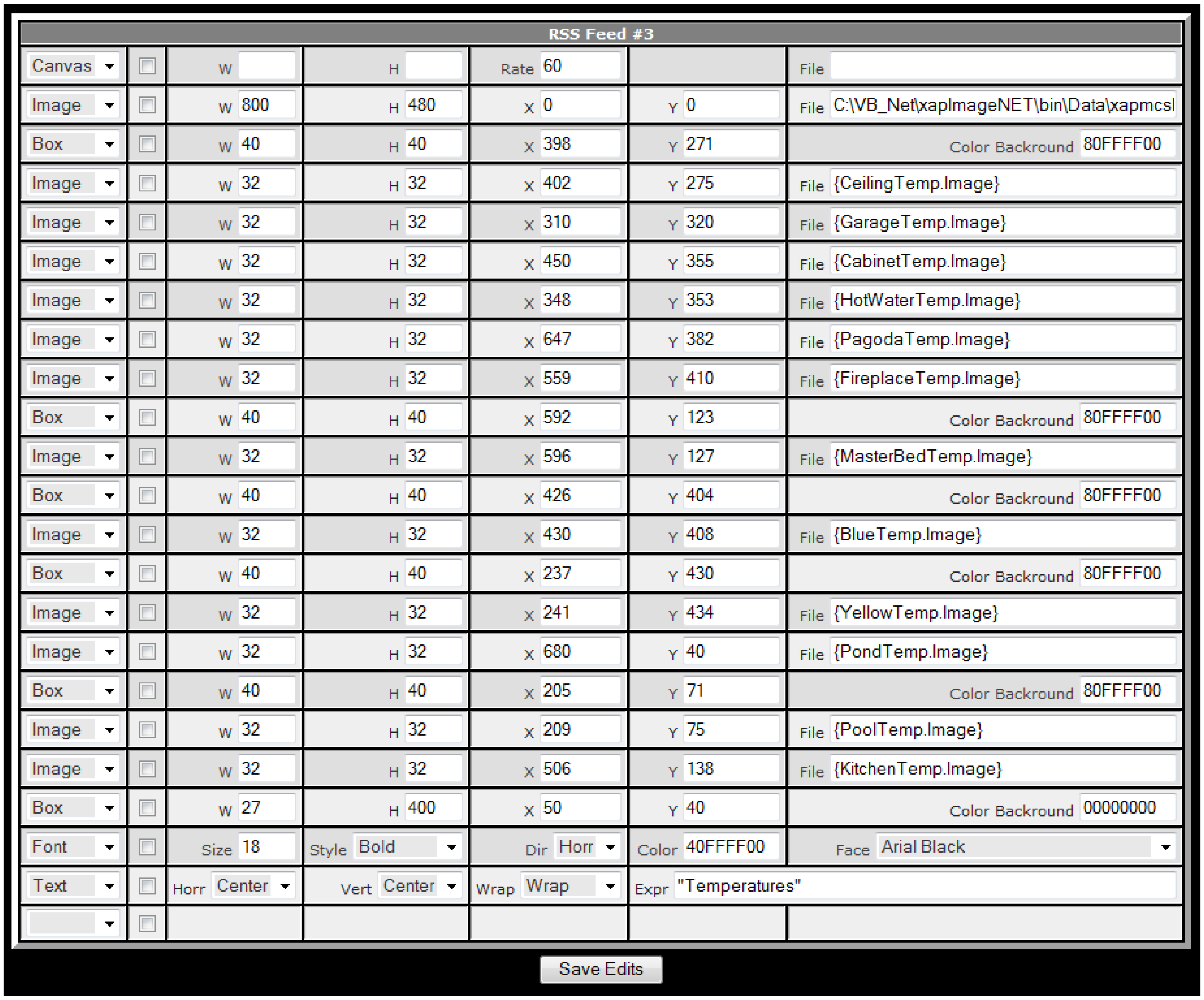
Figure 8 Cropped and Timestamped Radar Image

The third example illustrates how HTML icons can be used and text can be produced vertically. In this case a floorplan image is the base image upon which temperature measurement icons are overlayed. Some of the icons are encased in green boxes. This was done to distinguish first floor from second floor room measurements. A vertical text label was also added.

The raw icons that contain temperature readings are only 16x16 pixels. This was too small to be effectively viewable so they were enlarged to 32x32. This makes for a somewhat blurry icon. The other approach is to make new icons dynamically by using TEXT rather than IMAGE activities.

The room temperature reading icons were obtained using the ".Image" property of the dynamic data identified by an alias name. The X/Y positioning of the icons was selected by using another application that provides cursor feedback when viewing the background image. In my case Photoshop was used. No provision for interactive placement of icons or text is supported with xapmcsImage.

The vertical text in the left column was achieved by using the "Wrap" option in the TEXT property. The BOX width was selected such that it is wide enough for only one character.



RSS Feed #3

Canvas	☐	W	H	Rate 60		File
Image	☐	W 800	H 480	X 0	Y 0	File C:\VB_Net\xapImageNET\bin\Data\xapmcsl
Box	☐	W 40	H 40	X 398	Y 271	Color Backround 80FFFF00
Image	☐	W 32	H 32	X 402	Y 275	File {CeilingTemp.Image}
Image	☐	W 32	H 32	X 310	Y 320	File {GarageTemp.Image}
Image	☐	W 32	H 32	X 450	Y 355	File {CabinetTemp.Image}
Image	☐	W 32	H 32	X 348	Y 353	File {HotWaterTemp.Image}
Image	☐	W 32	H 32	X 647	Y 382	File {PagodaTemp.Image}
Image	☐	W 32	H 32	X 559	Y 410	File {FireplaceTemp.Image}
Box	☐	W 40	H 40	X 592	Y 123	Color Backround 80FFFF00
Image	☐	W 32	H 32	X 596	Y 127	File {MasterBedTemp.Image}
Box	☐	W 40	H 40	X 426	Y 404	Color Backround 80FFFF00
Image	☐	W 32	H 32	X 430	Y 408	File {BlueTemp.Image}
Box	☐	W 40	H 40	X 237	Y 430	Color Backround 80FFFF00
Image	☐	W 32	H 32	X 241	Y 434	File {YellowTemp.Image}
Image	☐	W 32	H 32	X 680	Y 40	File {PondTemp.Image}
Box	☐	W 40	H 40	X 205	Y 71	Color Backround 80FFFF00
Image	☐	W 32	H 32	X 209	Y 75	File {PoolTemp.Image}
Image	☐	W 32	H 32	X 506	Y 138	File {KitchenTemp.Image}
Box	☐	W 27	H 400	X 50	Y 40	Color Backround 00000000
Font	☐	Size 18	Style Bold	Dir Horr	Color 40FFFF00	Face Arial Black
Text	☐	Horr Center	Vert Center	Wrap Wrap	Expr "Temperatures"	
	☐					

Figure 9 RSS Feed 3 Definition Example

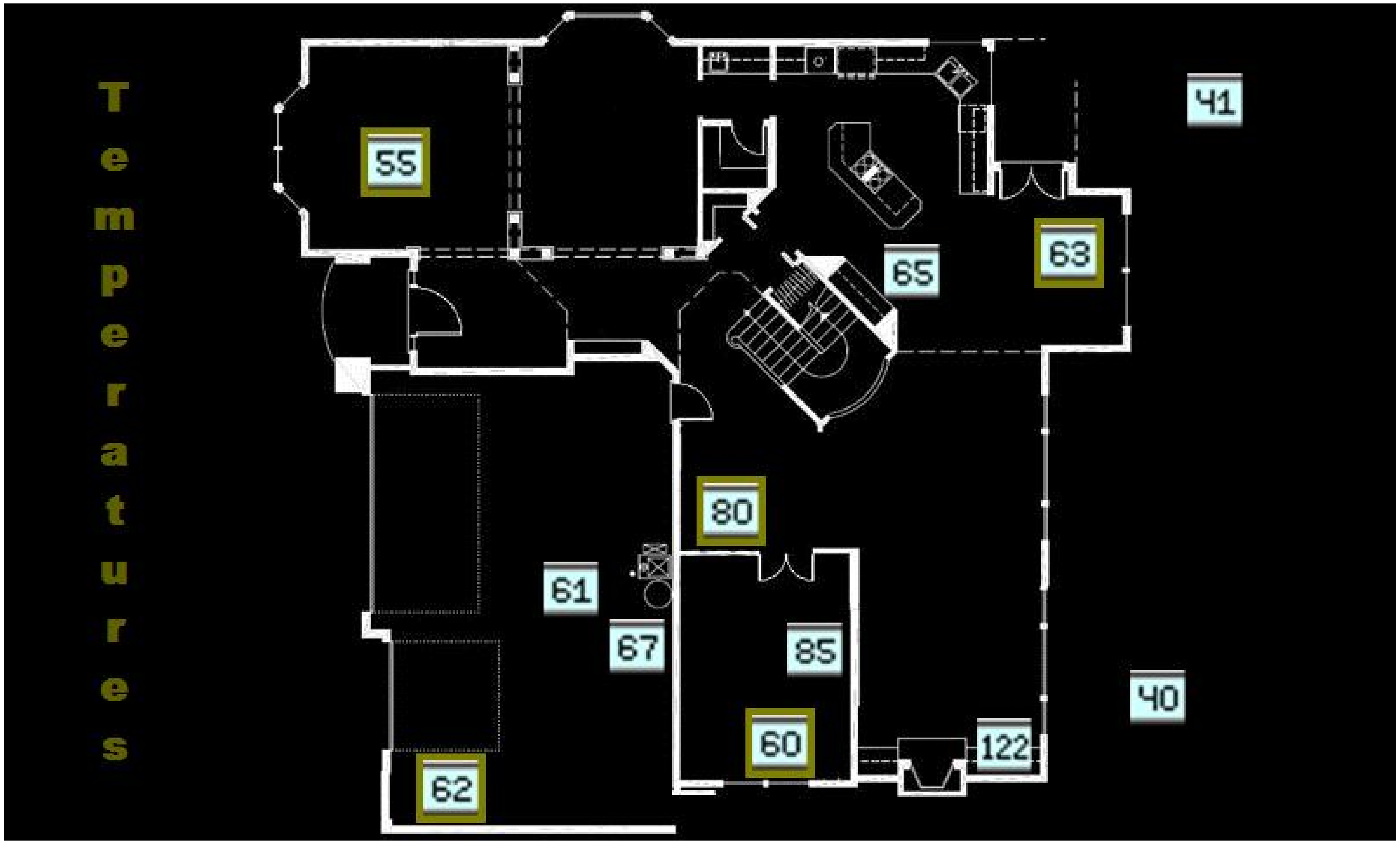
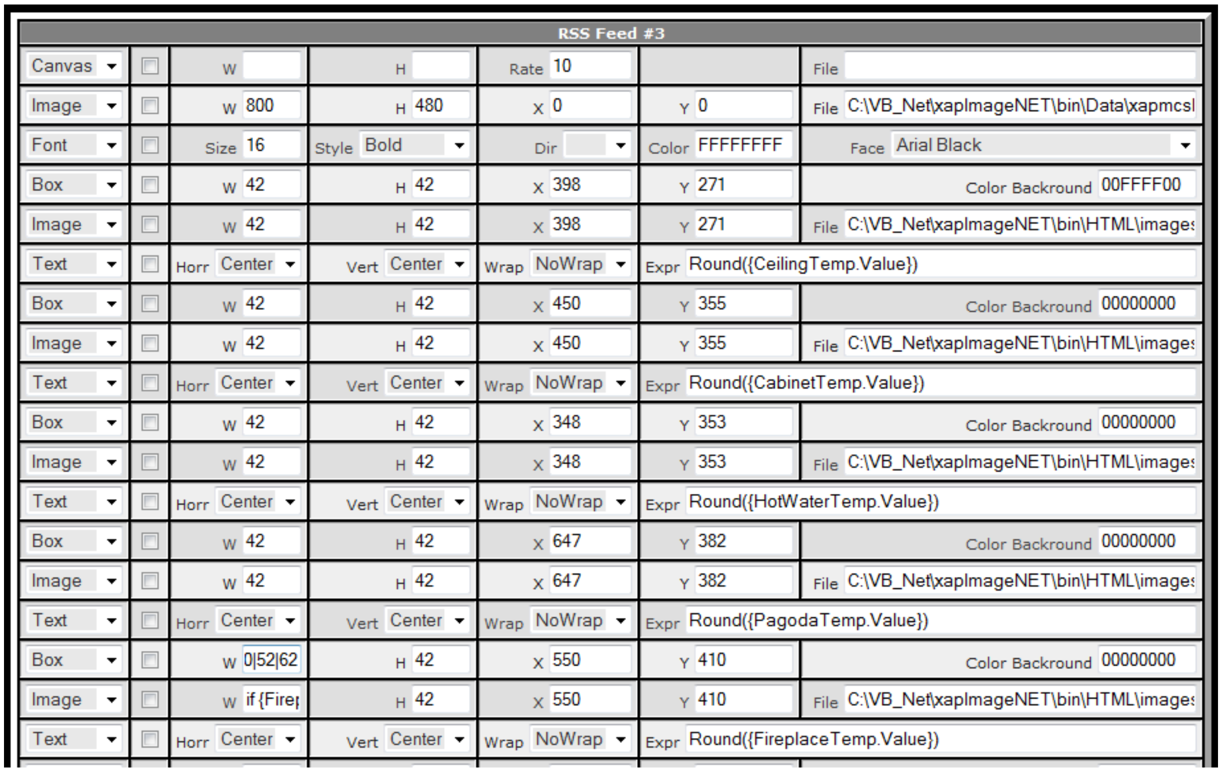


Figure 10 RSS Feed 3 Resultant Image

The resized icons in Figure 10 are usable, but more pleasing display can be achieved by overlaying text on a graphic button. This will eliminate the jaggedness resulting from upsizing a small icon. To accomplish this definition which is currently two lines each for each temperature reading will not be expanded to three lines. This adds the additional graphic button and replaces the Alias.Image with Alias.Value that will be used to produce the text value. The text will normally be two digits so a button and font appropriate for this length is selected. In some cases, however, the text is three characters and for that case the button needs to be enlarged to handle the larger text. This conditional sizing is done for the fireplace temperature using an If expression. Particular the Box and Image activities have a Width parameter that define the placement of text and this Width entry is made with the expression:

if {FireplaceTemp.Value}<100|52|62

This expression will evaluate to a width of 52 for temperatures lower than 100 degrees and 62 otherwise. One of three colored button are selected for the image to indicate location of temperature reading. These will be green, blue and red based upon first floor, second floor or outside. This is done by using a different graphic button image for each of the three. The definition segment and the resultant image are shown in Figure 11 and Figure 12.



RSS Feed #3

Canvas	☐	W	H	Rate 10		File
Image	☐	W 800	H 480	X 0	Y 0	File C:\VB_Net\xapImageNET\bin\Data\xapmcsl
Font	☐	Size 16	Style Bold	Dir	Color FFFFFFFF	Face Arial Black
Box	☐	W 42	H 42	X 398	Y 271	Color Backround 00FFFF00
Image	☐	W 42	H 42	X 398	Y 271	File C:\VB_Net\xapImageNET\bin\HTML\image
Text	☐	Horr Center	Vert Center	Wrap NoWrap	Expr Round({CeilingTemp.Value})	
Box	☐	W 42	H 42	X 450	Y 355	Color Backround 00000000
Image	☐	W 42	H 42	X 450	Y 355	File C:\VB_Net\xapImageNET\bin\HTML\image
Text	☐	Horr Center	Vert Center	Wrap NoWrap	Expr Round({CabinetTemp.Value})	
Box	☐	W 42	H 42	X 348	Y 353	Color Backround 00000000
Image	☐	W 42	H 42	X 348	Y 353	File C:\VB_Net\xapImageNET\bin\HTML\image
Text	☐	Horr Center	Vert Center	Wrap NoWrap	Expr Round({HotWaterTemp.Value})	
Box	☐	W 42	H 42	X 647	Y 382	Color Backround 00000000
Image	☐	W 42	H 42	X 647	Y 382	File C:\VB_Net\xapImageNET\bin\HTML\image
Text	☐	Horr Center	Vert Center	Wrap NoWrap	Expr Round({PagodaTemp.Value})	
Box	☐	W 0|52|62	H 42	X 550	Y 410	Color Backround 00000000
Image	☐	W if {Firep	H 42	X 550	Y 410	File C:\VB_Net\xapImageNET\bin\HTML\image
Text	☐	Horr Center	Vert Center	Wrap NoWrap	Expr Round({FireplaceTemp.Value})	

Figure 11 Conditional Expression and Text over Button Definition

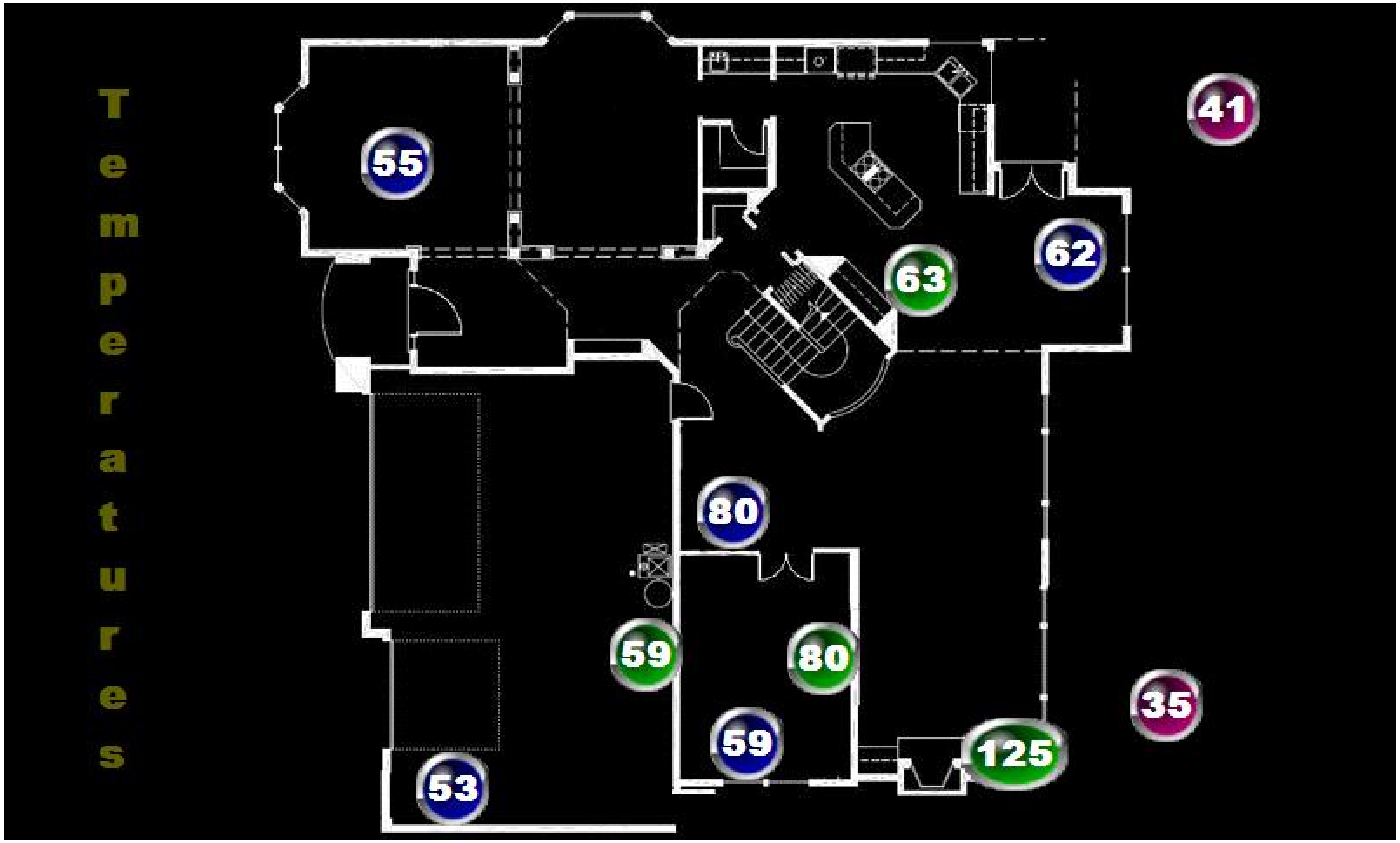


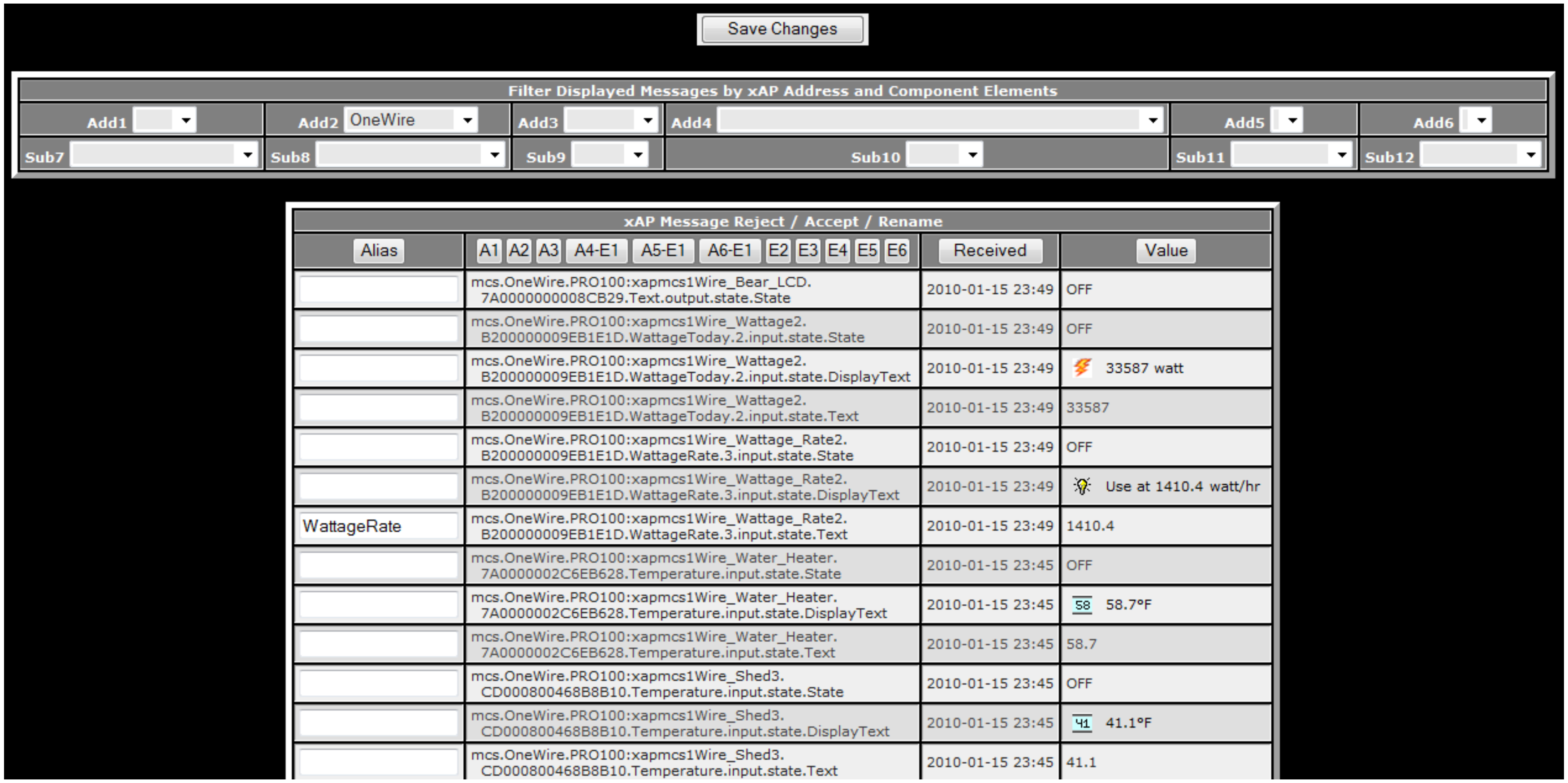
Figure 12 Image with Text over Dynamically Sized Buttons

4 xAP Data and Setup

All xAP transmitted xAP data is available for use by xapmcsImge. Any data that is to become part of a constructed image is identified with an Alias name. To assist in identifying the data of interest the most recently received values and time of receipt are shown. A filter mechanism is also available to show selected subsets of the data. A second filter mechanism is also available from the setup form at the bottom of the page where the display can be restricted to only data that has an alias name associated with it.

A message item is identified by its address, group, and key elements. The filter is provided in two rows and the address/group/key information of each item is also shown in two lines of each row of the table. The first row will be the Source address and either the first segment of the subaddress or the first element of the group name. The second row will contain all subsequent selection elements.

Figure 9 show a segment of the xAP Data page. In this example it shows only data where the second address element is “OneWire”. It shows one item selected with an Alias named “WattageRate”. In this case it is the Text=XXXX element of the specific message. When this Alias is used in the RSS Image page (i.e. {WattageRate}) then the text being used to support the image display will be 1410.4. This value will change over time depending upon more recent xAP messages from the same source.



Filter Displayed Messages by xAP Address and Component Elements

Add1	Add2 OneWire	Add3	Add4	Add5	Add6
Sub7	Sub8	Sub9	Sub10	Sub11	Sub12

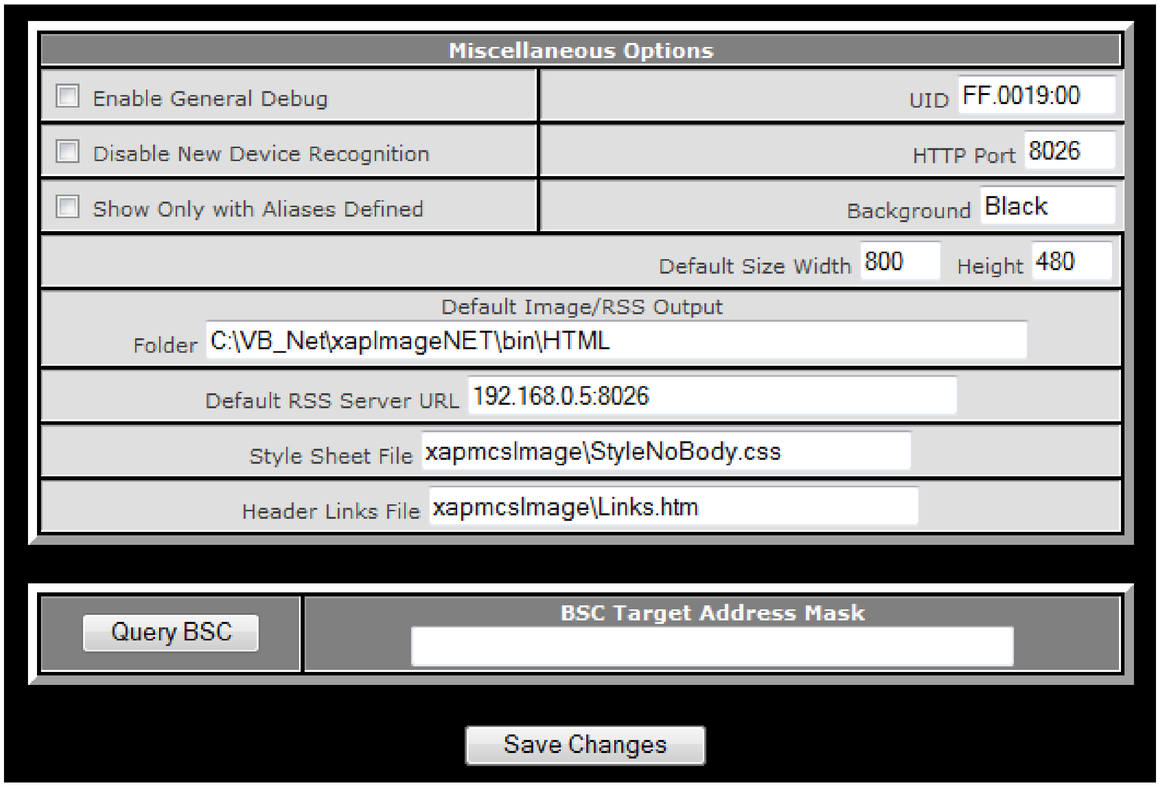
xAP Message Reject / Accept / Rename

Alias	A1 A2 A3 A4-E1 A5-E1 A6-E1 E2 E3 E4 E5 E6	Received	Value
	mcs.OneWire.PRO100:xapmcs1Wire_Bear_LCD. 7A0000000008CB29.Text.output.state.State	2010-01-15 23:49	OFF
	mcs.OneWire.PRO100:xapmcs1Wire_Wattage2. B200000009EB1E1D.WattageToday.2.input.state.State	2010-01-15 23:49	OFF
	mcs.OneWire.PRO100:xapmcs1Wire_Wattage2. B200000009EB1E1D.WattageToday.2.input.state.DisplayText	2010-01-15 23:49	33587 watt
	mcs.OneWire.PRO100:xapmcs1Wire_Wattage2. B200000009EB1E1D.WattageToday.2.input.state.Text	2010-01-15 23:49	33587
	mcs.OneWire.PRO100:xapmcs1Wire_Wattage_Rate2. B200000009EB1E1D.WattageRate.3.input.state.State	2010-01-15 23:49	OFF
	mcs.OneWire.PRO100:xapmcs1Wire_Wattage_Rate2. B200000009EB1E1D.WattageRate.3.input.state.DisplayText	2010-01-15 23:49	Use at 1410.4 watt/hr
WattageRate	mcs.OneWire.PRO100:xapmcs1Wire_Wattage_Rate2. B200000009EB1E1D.WattageRate.3.input.state.Text	2010-01-15 23:49	1410.4
	mcs.OneWire.PRO100:xapmcs1Wire_Water_Heater. 7A0000002C6EB628.Temperature.input.state.State	2010-01-15 23:45	OFF
	mcs.OneWire.PRO100:xapmcs1Wire_Water_Heater. 7A0000002C6EB628.Temperature.input.state.DisplayText	2010-01-15 23:45	58.7°F
	mcs.OneWire.PRO100:xapmcs1Wire_Water_Heater. 7A0000002C6EB628.Temperature.input.state.Text	2010-01-15 23:45	58.7
	mcs.OneWire.PRO100:xapmcs1Wire_Shed3. CD000800468B8B10.Temperature.input.state.State	2010-01-15 23:45	OFF
	mcs.OneWire.PRO100:xapmcs1Wire_Shed3. CD000800468B8B10.Temperature.input.state.DisplayText	2010-01-15 23:45	41.1°F
	mcs.OneWire.PRO100:xapmcs1Wire_Shed3. CD000800468B8B10.Temperature.input.state.Text	2010-01-15 23:45	41.1

Figure 13 xAP Alias Selection

A few setup options are available that primarily affect the user interface. The HTTP port, the style sheet used, the default background color and the links to be included at the top of the page can be changed. These are shown in Figure 14. The xAP interface parameters consist of the UID, a function to query BSC schema devices, and an option to disable recognition of new devices. The query and recognition will most often be used when dealing with new xAP hardware and software.

The default RSS server is xapmcsImage and the default location where the constructed images are built is the \HTML folder under the xapmcsImage install. If a different server or image location is desired such as the Homeseer server, then the default can be specified in this area of the setup. This will eliminate the need from including this information in the definition lines.



☐ Enable General Debug

☐ Disable New Device Recognition

☐ Show Only with Aliases Defined

UID FF.0019:00

HTTP Port 8026

Background Black

Default Size Width 800 Height 480

Default Image/RSS Output Folder C:\VB_Net\xapImageNET\bin\HTML

Default RSS Server URL 192.168.0.5:8026

Style Sheet File xapmcsImage\StyleNoBody.css

Header Links File xapmcsImage\Links.htm

Figure 14 xAP Data Setup Options

5 xAP Setup

This section describes the initial setup of a xAP environment for Homeseer. It provides a high level description of xAP, the basic setup of the mcsXap plugin, setup of an xAP hub using the xapmcsHub application and a brief introduction to the xAP Viewer.

xapmcsImage does not depend upon Homeseer, but if information from Homeseer is desired to be used in the constructed Digital Frame images then the mcsXap plugin is necessary.

If xAP message data is to be used in the construction of Digital Frame images then a xAP hub is needed on the same computer that xapmcsImage is installed. If xAP data will not be used as part of the image construction then a xAP hub is not required.

5.1 Background

xAP protocol is implemented as UDP datagrams and this provides some benefits and some shortcomings. UDP datagrams have a very small protocol overhead which means they are easy to encode and decode and consume a very small footprint for low overall network utilization. The data is broadcast which means every device on the network receives the same information simultaneously. This makes for a convenient distributed computing environment. xAP is designed so it can operate on the smallest of processors such as the PIC so application in dedicated embedded applications is quite achievable. There do exist hardware devices that natively communicate with xAP.

The tradeoff for UDP vs. TCP is that communications are not assured by the protocol. A UDP message is transmitted and there is no acknowledge of its receipt or retry if the transmission fails. Since messages are broadcasted every devices will receive it and process it through its recognition stack to the point of recognizing it is or is not interested in its content. UDP is also intended only for LAN and not WAN communications. It is possible to control remote locations with a TCP communication, but this cannot be done with an UDP one.

The xAP protocol layer on top of UDP does address some of the shortcomings. It provides a schema whereby message receipt acknowledgement is provided. Applications are also available that will wrap xAP messages inside of TCP communications to allow WAN control via xAP. In both of these cases the applications become responsible for communication integrity rather than having this responsibility at the transport layer.

mcsXap is a Homeseer plugin that is able to recognized LAN traffic using the xAP protocol. This protocol is just another one of the many that have been defined (e.g. SMTP, HTTP) and layered on top of the IP network.

The xAP protocol is extensible. There are core schema that provide for maximum interoperability and there are specialized one that target specific niche. mcsXap is able to recognize all xAP schema, but is only able to translate selected ones into the Homeseer model of devices.

mcsXap will decode and encode xAPBSC, xap-x10 and xap-IR schema which is widely used core schema. xAPBSC provides the ability to communicate Homeseer DeviceStatus, DeviceValue, and DeviceString. xap-x10 provides the ability for additional X10 interfaces to be connected to Homeseer. Xap-IR provides the ability for additional IR interfaces to be connected to Homeseer.

It also support schema that are developed around the mcs environment and includes setio, role, time, wdt, homeseer, voice, and writelog. These special schema have application within a mcs xAP environment, but will not have application to those outside of Homeseer-oriented setups. For example, the homeseer.event schema is used to allow the Homeseer Log to exist on the LAN rather than within Homeseer and all xAP applications can write to this log just as one would write to the Homeseer Log using hs.Writelog.

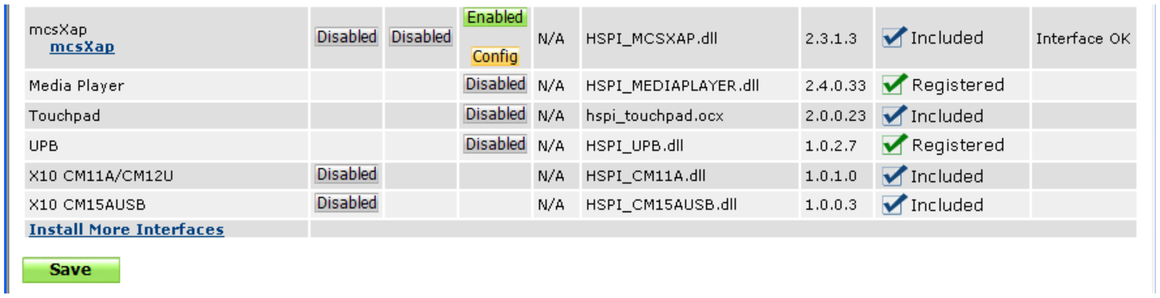
The typical initial use of xAP with Homeseer is to interface with some other xAP application such as xapmcs1wire to get 1-wire sensor readings into Homeseer or xapmcsRF to get W800 or RFXCOM RF translations into Homeseer. In most of these cases the information of interest is communicated using xAPBSC schema and that will be the only schema that needs to be considered within mcsXap.

5.2 mcsXap Installation

mcsXap is provided in two versions. One is targeted to Homeseer V1.x and is an .ocx implemented in VB6. The second is targeted to Homeseer V2.x and is a .dll implemented in VB.NET.

The set of files are obtained from the xAP Library section of the Homeseer Message Board in the form of a zip container. The contents of the zip file should be extracted into the same folder where Homeseer.exe is running. This will be something like C:\Program Files\Homeseer HS2\ but several variants are used.

The next time Homeseer starts it will recognize mcsXap and it will be available to be included as an active Homeseer plugin. Figure 11 shows a segment of the Setup\Interfaces page. mcsXap is shown as the first row. The three columns of buttons indicate that it can be used for IR, X10 and Other. For most users it will only be an Other interface with IR and X10 provided by other plugins. The selections in this figure show it being enabled only as an Other plugin. The Save button must be used to record this intent after the button is used to enable it.



mcsXap **mcsXap**	Disabled	Disabled	Enabled Config	N/A	HSPI_MCSXAP.dll	2.3.1.3	✓ Included	Interface OK
Media Player			Disabled	N/A	HSPI_MEDIAPLAYER.dll	2.4.0.33	✓ Registered	
Touchpad			Disabled	N/A	hspi_touchpad.ocx	2.0.0.23	✓ Included	
UPB			Disabled	N/A	HSPI_UPB.dll	1.0.2.7	✓ Registered	
X10 CM11A/CM12U	Disabled			N/A	HSPI_CM11A.dll	1.0.1.0	✓ Included	
X10 CM15AUSB	Disabled			N/A	HSPI_CM15AUSB.dll	1.0.0.3	✓ Included	
Install More Interfaces								

Figure 15 Homseer Setup Interfaces Page

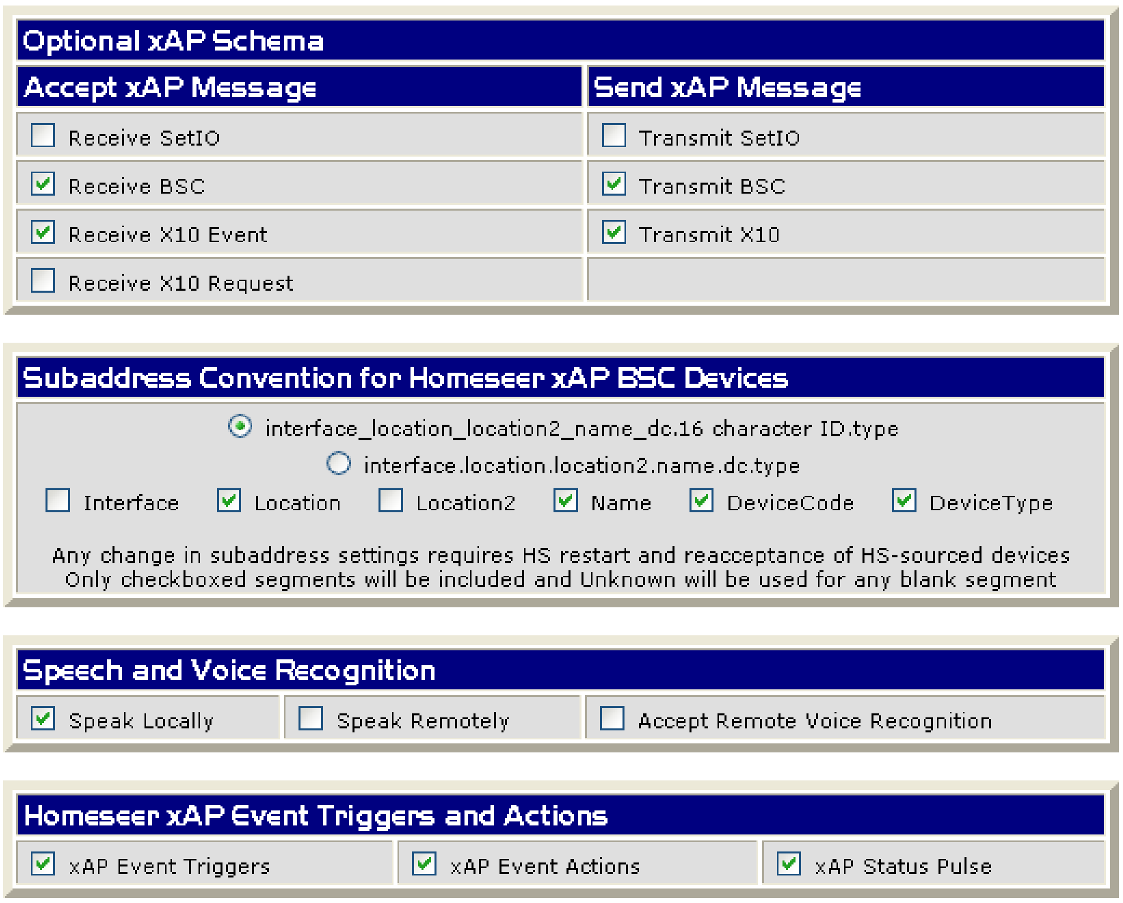
Once mcsXap is active as a Homeseer plugin then it needs to be setup to identify the scope of what it is suppose to do. mcsXap is primarily a conduit to translate information into a Homeseer view of it. It is just like the CM11A/CMU12U plugin that translates the X10 protocol on the powerline or the UPB plugin that translates UPB powerline protocol. In the mcsXap case the LAN is the physical connection and xAP is the protocol.

When mcsXap or any mcs xAP application starts it will look for the first NIC and assume that this NIC will be the one to use for xAP communications. If this default is not the correct one then a specific one can be identified by including the desired information in the file \Config\mcsXap.ini. A sample mcsXap.ini showing the setup for the NIC using address 192.168.0.117 is shown below.

```
[INTERFACE]
InterfaceAddress=192.168.0.117
BroadcastAddress=192.168.0.255
```

5.3 mcsXap Initial Setup

Figure 12 contains the settings of most interest in getting mcsXap functional. For most users the only xAP schema of interest will be xAPBSC and to enable the Receive BSC and Transmit BSC checkboxes will be checked. Figure 12 shows also the three X10 related options as being selected. Most users will not have these enabled, but no harm is one with them enabled.



Optional xAP Schema

Accept xAP Message	Send xAP Message
Receive SetIO	Transmit SetIO
[✓ Receive BSC]	[✓ Transmit BSC]
[✓ Receive X10 Event]	[✓ Transmit X10]
Receive X10 Request	

Subaddress Convention for Homeseer xAP BSC Devices

[✓ interface_location_location2_name_dc.16 character ID.type]
interface.location.location2.name.dc.type

Interface [✓ Location] Location2 [✓ Name] [✓ DeviceCode] [✓ DeviceType]

Any change in subaddress settings requires HS restart and reacceptance of HS-sourced devices
Only checkboxed segments will be included and Unknown will be used for any blank segment

Speech and Voice Recognition

[✓ Speak Locally] Speak Remotely Accept Remote Voice Recognition

Homeseer xAP Event Triggers and Actions

[✓ xAP Event Triggers] [✓ xAP Event Actions] [✓ xAP Status Pulse]

Figure 16 mcsXap Primary Setup Options

The second setup section shown in Figure 12 is used to define how the address of the xAPBSC message is constructed. The address is the mapping between a Homeseer device reference and an xAP message that will be recognized by other xAP applications. In general it does not matter what the address format is, but whatever is selected should not be changed once the xAP environment is established. A change in the xAP address would be similar to a wholesale change in all the device codes in Homeseer thus resulting in breakage of all the events that use devices.

The mcs xAP environment has a convention of Location_Name_DC.SerialNumber.Type for identification of Homeseer devices. This simply means that the default information extracted from the address can be used to automatically determine the device type, name, etc. The SerialNumber segment of the address is used by the database application within the mcs environment to define a database table or field and start collecting data about the device.

Other individuals have different conventions as to how they address devices. Two options are provided by mcsXap to accomplish the address definition. The first one forces a three segment address and the second one is a variable segment one. The checkboxes are used to select which information is included in each layout.

Unless there is a reason to change the default the default should be used to minimize downstream headaches.

Speech and Voice Recognition does not have the utility it once had before HS2.x and the speaker client. For most users none of the checkboxes should be checked in this section.

The Event Triggers and Actions will likely be of interest eventually, but are not needed initially. An event trigger can be setup to look for a specific xAP message. Since the messages that most users will be using are xAPBSC and these messages are mapped into Homeseer devices, then a standard HS event trigger on a device will be the easiest way to recognize the event.

For the same reason the Action to send an xAP message will also not be used by most since the messages they will be sending will be xAPBSC and this is sent by simply changing the value/status/string of the Homeseer device.

What some may find useful is the Event Trigger based upon a lack of data being updated by an external xAP application. This would then be used to take an action to restart the application or generate message to provide awareness.

For the initial setup the Event support within mcsXap should not bog one down. It can always be added in the future when a need is identified.

There are several other setup options available and these should all be left in the disabled or default state. When specific need exists and greater comfort exists in the role that mcsXap provides then it makes sense to revisit the other options available.

5.4 mcsXap Initial Use

The first thing that one needs to do is get visibility into xAP traffic that exists. There are three elements to this. One is the xAP Hub. The second is the xAP Viewer. The third is the mcsXap viewing filters.

When a UDP datagram is sent it must be directed to a specific port or socket. xAP has reserved port 3639 for this purpose. It is much like HTTP defaults to 80 and SMTP defaults to 25 etc. Just as you cannot have two Web servers both responding to port 80, you cannot have more than one xAP application responding to port 3639. This is where the xAP hub comes into play. The xAP hub is given the responsibility to listen for all incoming xAP messages on port 3639. When it gets one it uses the localhost (127.0.0.0) interface to retransmit that message to every other xAP application running on the same computer. When something like xapmcs1Wire broadcasts a sensor value change that broadcasted message is actually received on the computer by the xAP hub and the xAP hub retransmits it locally to mcsXap and all others running on that same computer.

Since the xAP hub is the router for xAP traffic it is desirable that it is running before other xAP applications are started. This way startup information from the other applications will be routed to others that may want to listen.

The second item is the xAP Viewer. The xAP Viewer is a diagnostic tool. It is not required for normal xAP operations, but when some investigation is needed then it is an essential tool. It shows all the xAP traffic and presents it in an easily viewable manner.

The xAP viewer is a hybrid application. If it starts and it recognizes that there is nothing providing the role of xAP Hub then it will assume this role. If it does have this role (as is indicated in its title bar) and the viewer is closed then nothing will be serving the role of xAP hub and communications will effectively stop, but while it is open there will be communications.

The third item is within mcsXap with setup as shown in Figure 13. At the bottom of this figure are two lines that each show an xAP message which has the potential to be mapped into an Homeseer device. If the "A"ccept column checkbox is checked then all future receptions of a message from this address will be decoded and the values placed in the DeviceStatus, DeviceValue and DeviceString of the Homeseer Device.

At the time the "A" checkbox is checked a specific Device Code can be entered into text box on the same row. There is also a location and a name textbox on the same line that are not visible on the inserted figure. If all three of these are left blank then mcsXap will assign a Device Code and give it a name and location. These three parameters can be later changed without implications to the functioning of mcsXap. If changes are made it is best to do if from mcsXap rather than from Homeseer so mcsXap will know about the change.

There are five checkboxes at the top portion of Figure 13. These are used to identify the segment of the population of received xAP messages that should be viewed on the browser on the lines that were discussed in the prior paragraph.

The "Show Homeseer Received BSC" will normally be checked since xAPBSC schema is the one most used and visibility into what is received will be desired.

The "Show Other Received xAP Schema" covers all other messages. These messages are referred to as "Raw". For these messages mcsXap will map their contents into Homeseer devices as best it can. This usually means the message body will be place in the DeviceString. What this means is that events based upon new values from xAP messages will not be able to be setup using the event model of Homeseer which triggers only on DeviceStatus and DeviceValue. If changes in these devices need event notification then the mcsXap Event Trigger can be used. This, however, is typically not needed since the message content is generally informative rather than actionable.

Every device in Homeseer can have its values sent as xAP messages using the xAPBSC schema. It is likely that a limited number of the devices will need to be transmitted for other xAP applications to use. This of course depends upon the extent of the xAP network. The “Show Homeseer Sourced BSC” checkbox will make the set of device available on this page and then “A”ccept checkbox used to select the device as being active as an xAP message. The message will be sent every time Homeseer changes the DeviceStatus, DeviceValue or DeviceString of the device. It will also be sent when queries using the xAP query schema.

The query schema is xAPBSC.Query. xAP allows wildcarding with xAPBSC as well as all other xAP schema. That means that a query can be made for the status of a single device or of a class of multiple devices.

mcsXap also provides a button to simulate a fully wildcarded xAPBSC.Query message. This is “Send BSC” button. When this button is clicked then the values for all Homeseer devices that have been “A”ccepted for xAP message transmission will be sent using the xAPBSC.Info schema.

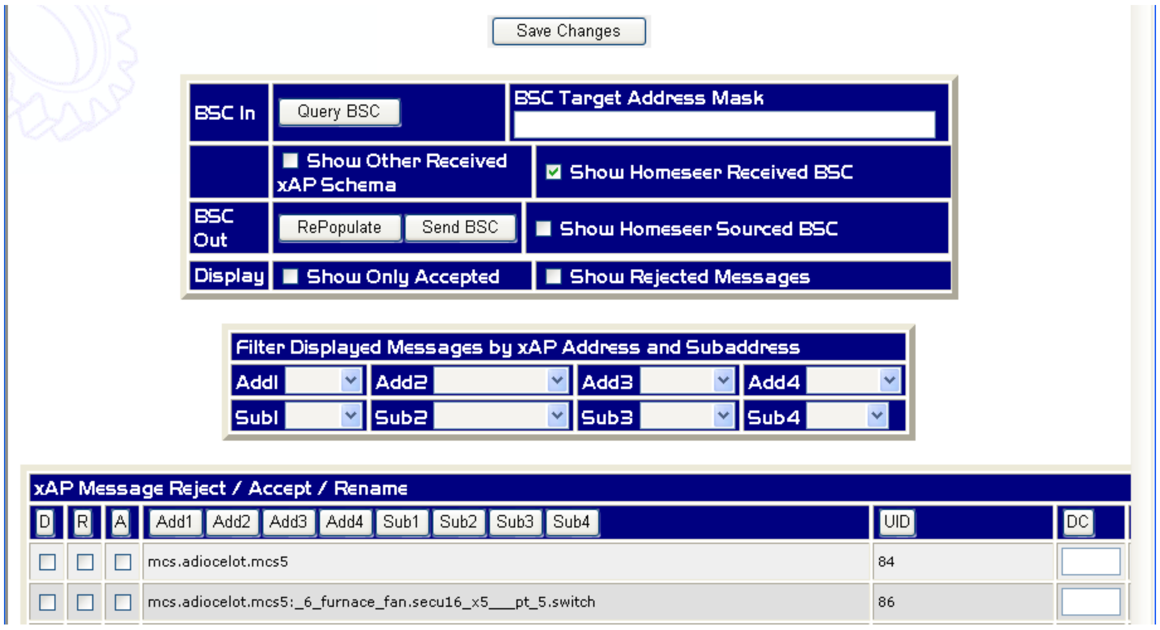
There is also a “Query BSC” button on the line labeled “BSC In”. When this button is clicked a fully wildcarded xAPBSC.Query message will be sent to all other xAP applications on the LAN and each application will respond with the current values of all their devices. When Homeseer starts mcsXap will automatically send this same request so it becomes synchronized with the entire xAP network.

The “BSC In” line also has input provisions for “BSC Target Address Mask”. When this is filled in then the scope will be reduce of the xAP endpoints that will respond. The mask can be a specific end point or partially wildcarded. Initially a fully wildcarded query will suffice. When there is more familiarity with the xAP addressesing structure then more restrictive queries can be used.

If new devices are added to Homeseer outside of mcsXap and this device is to be transmitted via xAP then it will be necessary to use the “Repopulate” button on the “BSC Out” line. mcsXap populates its image of Homeseer devices at startup so a restart will also have the same effect as using the “Repopulate” button.

The “Display” line checkboxes are used as filters based upon the “R”ejected and “A”ccepted checkboxes associated with each device in the body of the table. Normally the “Show Only Accepted” is the only one that is checked for normal viewing. When new devices are to be added it is unchecked so they become visible for selection.

The “R”ejected checkbox is used a display filtering mechanism to hide devices that are not of immediate interest. There is also a “D”elete checkbox. When it is used then all record of receiving the message will be removed. It is appropriate for invalid messages of for messages that were received from hardware that no longer exists.



BSC In	Query BSC	BSC Target Address Mask
	☐ Show Other Received xAP Schema	☑ Show Homeseer Received BSC
BSC Out	RePopulate / Send BSC	☐ Show Homeseer Sourced BSC
Display	☐ Show Only Accepted	☐ Show Rejected Messages

Filter Displayed Messages by xAP Address and Subaddress

Add1	Add2	Add3	Add4
Sub1	Sub2	Sub3	Sub4

xAP Message Reject / Accept / Rename

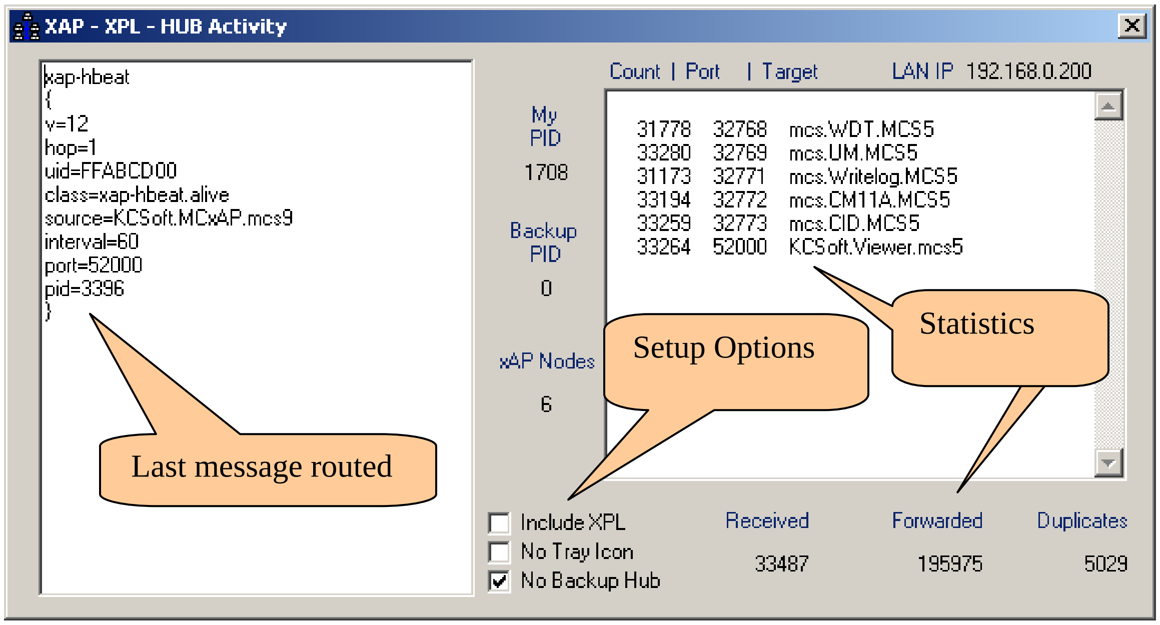
D	R	A	Add1 Add2 Add3 Add4 Sub1 Sub2 Sub3 Sub4	UID	DC
☐	☐	☐	mcs.adiocelot.mcs5	84	
☐	☐	☐	mcs.adiocelot.mcs5:_6_furnace_fan.secu16_x5___pt_5.switch	86	

Figure 17 Message Selection Filters

5.5 xapmcsHub Setup

xapmcsHub is available from the xAP Library section of the Homeseer Message Board. It is a VB6 .exe application that provides a user interface access via a tray icon. xapmcsHub should be installed in its own folder. A structure such as C:\Program Files\xAP\xapmcsHub\xapmcsHub.exe is appropriate. Other xAP applications would also be installed in a similar manner in independent folders.

The setup form is viewed from the tray icon. Three checkboxes are provided for setup of xapmcsHub. The typical configuration will be as shown in Figure 14 where only xAP will be processed, an icon will appear in the tray and no backup hub function will be spawned. Once stable operations are achieved the tray icon may be removed if one desires a less-cluttered tray.



```
xap-hbeat
{
v=12
hop=1
uid=FFABCD00
class=xap-hbeat.alive
source=KCSoft.MCxAP.mcs9
interval=60
port=52000
pid=3396
}
```

LAN IP 192.168.0.200

My PID 1708

Backup PID 0

xAP Nodes 6

```
Count   Port   Target
31778   32768  mcs.WDT.MCS5
33280   32769  mcs.UM.MCS5
31173   32771  mcs.Writelog.MCS5
33194   32772  mcs.CM11A.MCS5
33259   32773  mcs.CID.MCS5
33264   52000  KCSoft.Viewer.mcs5
```

- [ ] Include XPL
- [ ] No Tray Icon
- [x] No Backup Hub

Received	Forwarded	Duplicates
33487	195975	5029

Figure 18 xAP Hub Activity Form

There are a variety of xAP hubs available and any should be able to be used. There xFx (via xapautomation.org) makes available a .NET hub that can be installed as a service. This is an attractive option as it provides the benefit of running before other desktop xAP applications.

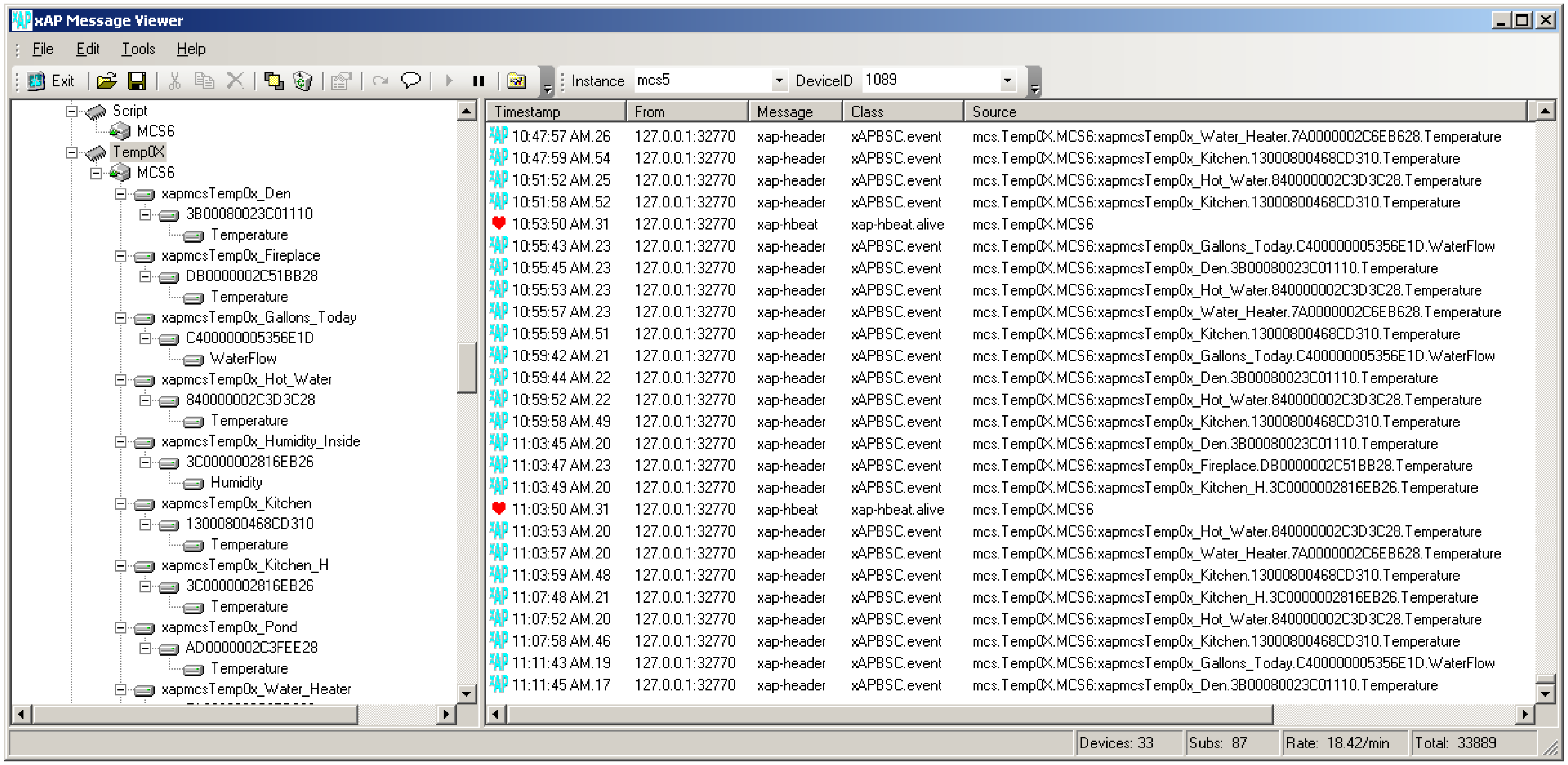
The only issue I have with it is its lack or removal of duplicate messages. In many case duplicates do not matter, but in cases where the message triggers events then having multiple events triggers is problematic. Note in Figure 14 that 5029 duplicate messages were detected and filtered-out by xapmcsHub.

5.6 xAP Message Viewer

The xAP Viewer was originally developed by xapFramework.net and is now maintained by xFx. It is a useful tool to observe xAP message traffic and it can also serve the dual role as a xAP hub.

Figure 15 and Figure 16 show a typical display for messages summaries and message detail that is available from a Window's GUI. When using the xAP viewer and a dedicated hub one needs to be careful to start the dedicated hub before the viewer otherwise the hub will not be able to gain access to the xAP port on the primary interface.

The left panel of the viewer provides a structured list of xAP endpoints. The + and – symbols can be used the compress and expand the structure. Clicking on any level will limit the contents of the right panel to only those endpoints. The right panel provides a chronological list of xAP messages. Clicking on a message from this panel will bring up a popup form that contains the detail of the message. An example is shown in Figure 16. The content of this second panel can be edited and it can be retransmitted. These features make it useful for diagnostic and testing activities.



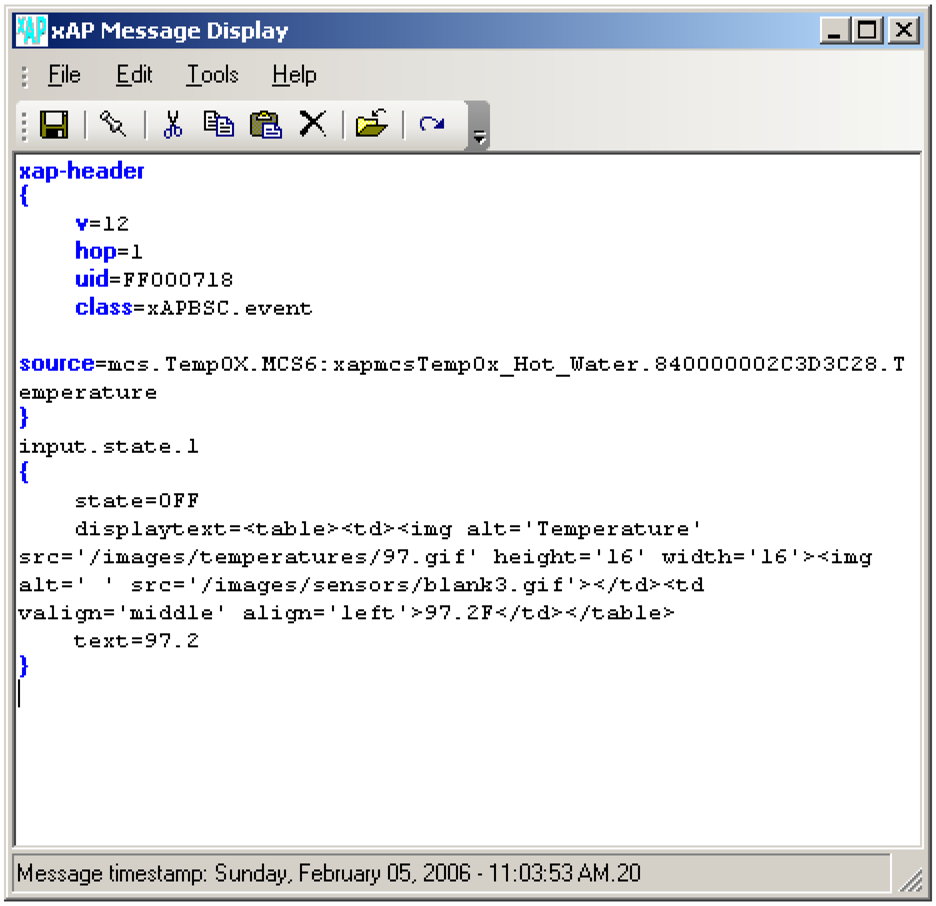
xAP Message Viewer

```
Script
    MCS6
Temp0X
    MCS6
        xapmcsTemp0x_Den
            3B00080023C01110
                Temperature
        xapmcsTemp0x_Fireplace
            DB0000002C51BB28
                Temperature
        xapmcsTemp0x_Gallons_Today
            C400000005356E1D
                WaterFlow
        xapmcsTemp0x_Hot_Water
            840000002C3D3C28
                Temperature
        xapmcsTemp0x_Humidity_Inside
            3C0000002816EB26
                Humidity
        xapmcsTemp0x_Kitchen
            13000800468CD310
                Temperature
        xapmcsTemp0x_Kitchen_H
            3C0000002816EB26
                Temperature
        xapmcsTemp0x_Pond
            AD0000002C3FEE28
                Temperature
        xapmcsTemp0x_Water_Heater
```

Timestamp	From	Message	Class	Source
10:47:57 AM.26	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Water_Heater.7A0000002C6EB628.Temperature
10:47:59 AM.54	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Kitchen.13000800468CD310.Temperature
10:51:52 AM.25	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Hot_Water.840000002C3D3C28.Temperature
10:51:58 AM.52	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Kitchen.13000800468CD310.Temperature
♥ 10:53:50 AM.31	127.0.0.1:32770	xap-hbeat	xap-hbeat.alive	mcs.Temp0X.MCS6
10:55:43 AM.23	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Gallons_Today.C400000005356E1D.WaterFlow
10:55:45 AM.23	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Den.3B00080023C01110.Temperature
10:55:53 AM.23	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Hot_Water.840000002C3D3C28.Temperature
10:55:57 AM.23	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Water_Heater.7A0000002C6EB628.Temperature
10:55:59 AM.51	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Kitchen.13000800468CD310.Temperature
10:59:42 AM.21	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Gallons_Today.C400000005356E1D.WaterFlow
10:59:44 AM.22	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Den.3B00080023C01110.Temperature
10:59:52 AM.22	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Hot_Water.840000002C3D3C28.Temperature
10:59:58 AM.49	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Kitchen.13000800468CD310.Temperature
11:03:45 AM.20	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Den.3B00080023C01110.Temperature
11:03:47 AM.23	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Fireplace.DB0000002C51BB28.Temperature
11:03:49 AM.20	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Kitchen_H.3C0000002816EB26.Temperature
♥ 11:03:50 AM.31	127.0.0.1:32770	xap-hbeat	xap-hbeat.alive	mcs.Temp0X.MCS6
11:03:53 AM.20	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Hot_Water.840000002C3D3C28.Temperature
11:03:57 AM.20	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Water_Heater.7A0000002C6EB628.Temperature
11:03:59 AM.48	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Kitchen.13000800468CD310.Temperature
11:07:48 AM.21	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Kitchen_H.3C0000002816EB26.Temperature
11:07:52 AM.20	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Hot_Water.840000002C3D3C28.Temperature
11:07:58 AM.46	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Kitchen.13000800468CD310.Temperature
11:11:43 AM.19	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Gallons_Today.C400000005356E1D.WaterFlow
11:11:45 AM.17	127.0.0.1:32770	xap-header	xAPBSC.event	mcs.Temp0X.MCS6:xapmcsTemp0x_Den.3B00080023C01110.Temperature

Devices: 33 | Subs: 87 | Rate: 18.42/min | Total: 33889

Figure 19 xAP Viewer



```
xap-header
{
    v=12
    hop=1
    uid=FF000718
    class=xAPBSC.event
source=mcs.TempOX.MCS6:xapmcsTemp0x_Hot_Water.840000002C3D3C28.T
emperature
}
input.state.1
{
    state=OFF
    displaytext=<table><td><img alt='Temperature'
src='/images/temperatures/97.gif' height='16' width='16'><img
alt=' ' src='/images/sensors/blank3.gif'></td><td
valign='middle' align='left'>97.2F</td></table>
    text=97.2
}
```

Message timestamp: Sunday, February 05, 2006 - 11:03:53 AM.20

Figure 20 xAP Viewer Message Detail

6 xAP Schema

The xAP schema described in this section is used by xapmcsImage.

6.1 xapmcsImage

6.1.1 Receive xAP Schema

All xAP messages are accepted

6.1.2 Transmit xAP Schema

```
WDT.Echo
        Echo.Response
                Response=<value received in wdt.echo/echo.query>

Homeseer.Role
        Role.Notification
                Role=Computer Management
                Path=<folder path to xapmcsWebControl.exe>

Homeseer.Event
        Event.Log
                Time=<time>
                Type=<text>
                Data=<text>
```