# xAPSonar

## Maxbotix
## MaxSonar Ultrasonic Range Finder

## Distributed Computing Node

Michael McSharry
July 13, 2009

## Table of Contents



## Table of Figures

# 1 Introduction

xapmcsSonar is a connector that bridges the Maxbotix product line of Ultrasonic Range Finders to xAP on an IP interface. The primary xAP schema utilized is Basic Status and Control 1.3. It is installed at any location on a PC and will use subfolders to retain configuration information, this document, and HTML formatting information.

This xAP node allows xAP message control and reporting of the hardware to provide distance measurement readings. It also allows for scripting based upon content of received messages or distance measurements. Signal processing control options are provided to best suit the nature of the application.

The Range Finder is modeled as single xAP node with an interface to a RS-232 port. The port provides input of distance measurements and two discrete outputs to control ranging and power. The xAP reporting uses xapbsc.event messages whenever the change in distance exceeds a user-setup hysteresis threshold. The Text and DisplayText keys are used to deliver the distance and HTML-formatted distance respectively. It also responds to xapbsc.query with a xapbsc.info message with the most recent distance measurement. xapbsc.cmd is used to control the power to the range finder hardware and this power-state is reflected in the state key of all xapbsc messages. It also responds to xapbsc.query by providing the current status.

The xapbsc subaddress is formed from three parts. The first is a user-provided location. The second is a unique name MaxbotixSONAR_# where # is the COM port number to which the range finder hardware is interfaced. The third is the type name “distance” of the node.

A fully formed typical xAP message is shown in section 1.1. Note the subaddress and UID as described above.

## *1.1 xAP Message Format*

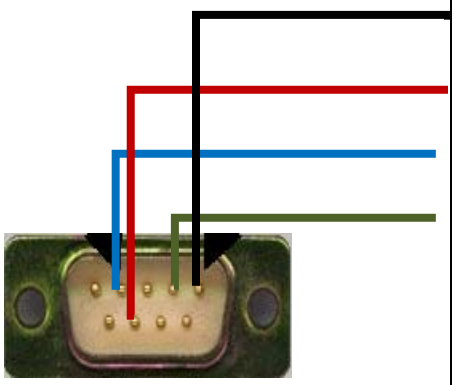
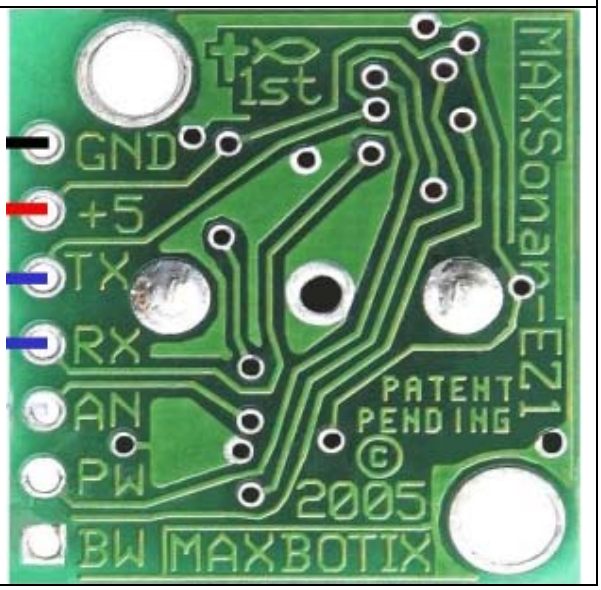
```
xap-header
{
v=13
hop=1
uid=FF.0067:0001
class=xapbsc.event
source=mcs.Sonar.MCS5:Den.MaxbotixSONAR__3.Distance
}
input.state
{
State=ON
DisplayText=<table><td><img alt='56.74' height='16' src='/xapmcsSonar/images/sensors/motion-on.gif'><img alt =
' ' src='/xapmcsSonar/images/sensors/blank1.gif'>56.74 in</td></table>
Text=56.74
}
```

## 2 Hardware Interface

The design interface is to a serial COM port of a PC where the voltage levels are nominally 0 to 5 VDC. Do not use a RS-232 interface that complies with the +/- 12VDC voltage swings as the MaxSonar is rated to a maximum of 5 Volts.

Power is provided from the PC via RTS Pin 7. Control of the ranging/transmitter is from PC via DTR Pin 4. Distance at 9600 baud is input to PC on RX Pin 2. Ground Pin 5 is used as common return for all lines.

Should the PC power source be noisy then a RC filter (100uf/100ohm) can be added between PC DB-9 RTS and the Maxsonar +5.

| DB-9 | MaxSonar |
|---|---|
| 1 N/C | 1 GND |
| 2 RX | 2 +5 |
| 3 N/C | 3 TX |
| 4 DTR | 4 RX |
| 5 GND | |
| 6 N/C | 5 AN |
| 7 RTS | 6 PW |
| 8 N/C | 7 BW |
| 9 N/C | |
|  |  |

# 8 Quick Start

The following outlines the steps to achieve computer control of the Sonar sensor via xapmcsSonar and Homeseer. The Homeseer-related steps are not required as xapmcsSonar is a self-sufficient computing node for a pure distributed environment.

- HOMESEER (V1)
  - Expand mcsXap plugin into the Homeseer folder
  - Register hspi_mcsXap.ocx using regsvr32 hspi_mcsXap.ocx in the Homeseer folder
  - Start Homeseer and select mcsXap as an IO interface
  - Configure mcsXap via its setup to use BSC transmit and receive
- XAP
  - Start an xAP hub such as xapmcsHub
  - Expand xapmcsSonar using the folder structure encoded in the zip file
  - Configure xapmcsSonar from tray icon to select the serial port and other preferences
  - Refresh browser to view the Sonar nodes
- HOMESEER INTEGRATION
  - Click the BSC Query of Receiver button to make the devices visible to mcsXap plugin
  - From mcsXap plugin browser accept message from xapmcsSonar

The above sequence is one of several possible configurations. Editing and customizations will naturally be peformed to tailor the setup as one's understanding improves.

# 9 Tray Interface

xapmcsSonar runs in the Windows tray with access via an icon that looks like a striped inverted triangle. This icon will blink for each ranging period of the hardware. Left of right click of the icon will present several options.

Exit is used to terminate xapmcsSonar.

The IO Window is used to see the raw traffic between the Maxbotix hardware and the PC.

The user interface access is from the top of menu options. The setup information can be set with the Browser interface menu option. If browser access has not yet been achieved then the \Config\xapmcsSonar.ini file can be edited.

The GUI, system config and status selections have not been activated in xapmcsSonar. They can be used to cancel the presence of the icon popup menu.

# 10 Browser Interface

## *10.1 Content Selection*

The browser interface provides administrative access as well as direct status and control of the features of the Sonar hardware. The setup is shown in Figure 1.

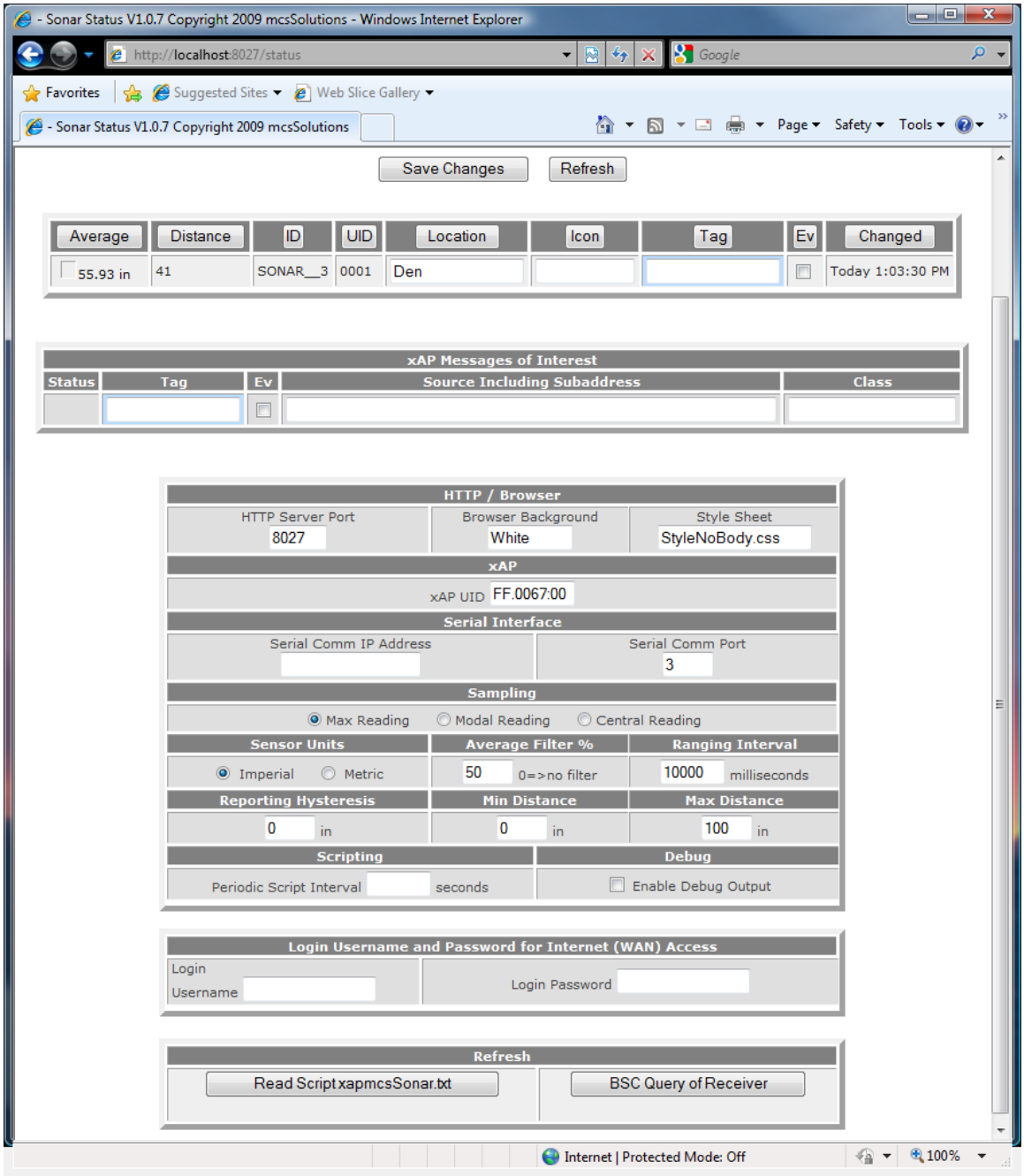



**Figure 1 Browser Content Selection**

## 10.2 Configuration

xapmcsSonar contains an HTTP server to provide the primary user interface. The port used by the server defaults to 8027, but can be changed via user input.

Most formatting is controlled by a style sheet except the body background. The default style sheet is the provided StyleNoBody.css located in the \HMTL\xapmcsSonar folder. This same folder contains icons as well as a file links.htm that contains links to other locations. These links are displayed under the Title bar of the browser page. The links.htm file can be deleted or edited to suit individual preferences.

Connection to the Sonar hardware is via a serial port. This port can be directly connected to the PC or connected via an IP. If directly connected then the port number is entered. If IP connected then both the IP address and port are entered. Note that power and ranging are controlled by discrete outputs of the serial interface so if an IP connection is used this control will only be available if this hardware and protocol are supported on this connection.

Each xAP node has a unique ID. xapmcsSonar defaults to FF.0067:00, but this can be changed as desired.

A beam is pulsed and readings taken for a period of about ten samples. From this set of ten there will be variation in the distance measurement and a selection from this set is to be made. Three approaches are available. These are largest value, most often occurring value or largest remaining sample after the largest has been discarded. The most recent selected sample is shown in the Distance column at the top of the browser display.

Each selection will then be averaged with prior readings. For a fast/noisy filter a filtering percentage near 0% should be selected. For a slow/smooth output a value near 99% should be used. This filtered output is the value transmitted in xAP messages as the current distance measurement. It is also shown at the top of the browser display.

Further signal processing can be done by discarding samples that outside an expected range. This range is entered as the min and max distances. Any reading outside this range will not be considered.

The xapbsc.event messages are sent whenever the change in distance since the last sent event exceeds a hysteresis threshold. This reduces the traffic on the LAN and subsequent processing requirements of the receiving nodes. A value of 0 will result in all selections being delivered. A large value will result in very few events. Current values can be requested at any time with the xapbsc.query message.

The thresholds and the distances will be in either English/Imperial or Metric units. One unit system is used for all values.

A script can be run based upon a valve change, selected xAP message reception, or periodically. To cause \Scripts\xapmcsSonar.txt to be executed periodically a number is entered into the Periodic Script Interval text box. The script will be called with no parameters in this case. Use of the script provides for local special case processing, but otherwise is not necessary.

Debug option is available to get more insight into internal logic. The debug file is produced in the \Data subfolder. The IO Window from the tray icon also provides a debug aid to view the data received from the hardware over the RS-232 link.

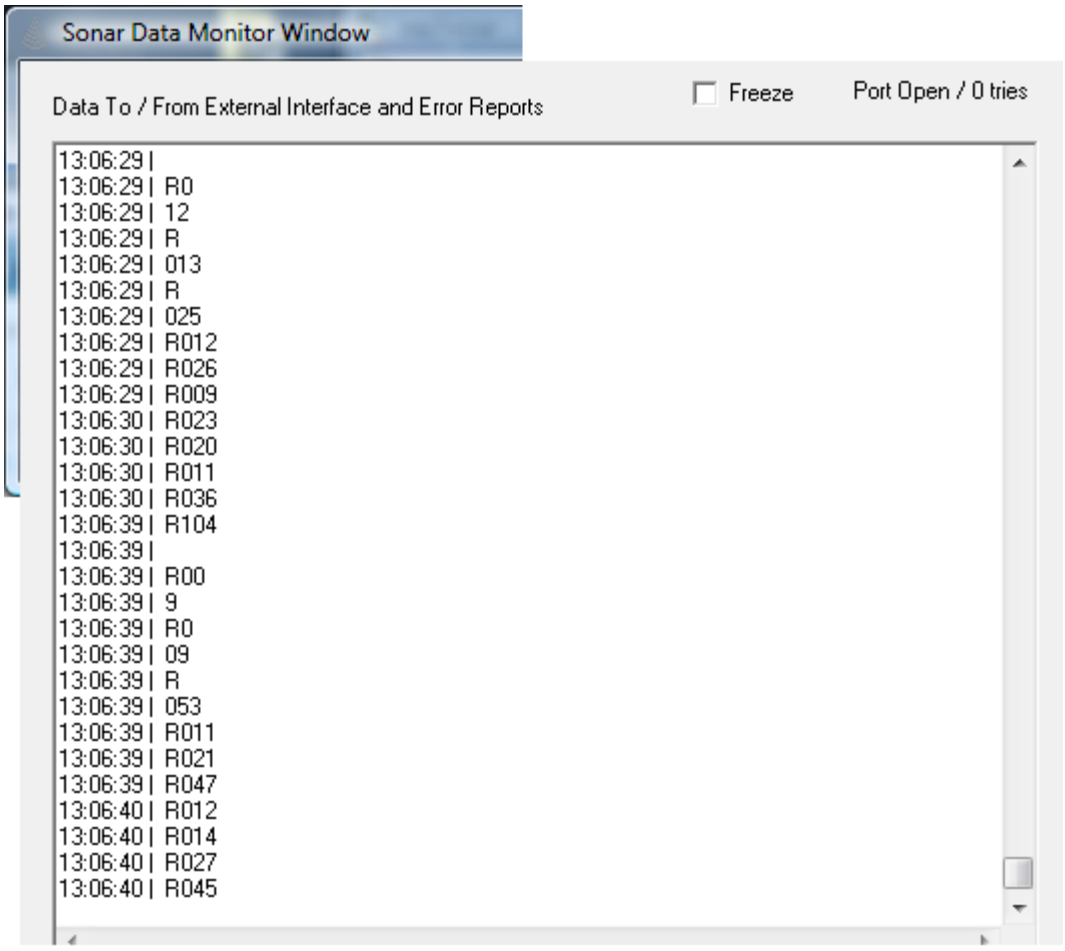



**Figure 2 IO Window**

## 10.3 Sonar Range Finder Device Properties

The “Location” entry is used to provide a name to the distance mesaaurement. It will include in the xAP message subaddress field.

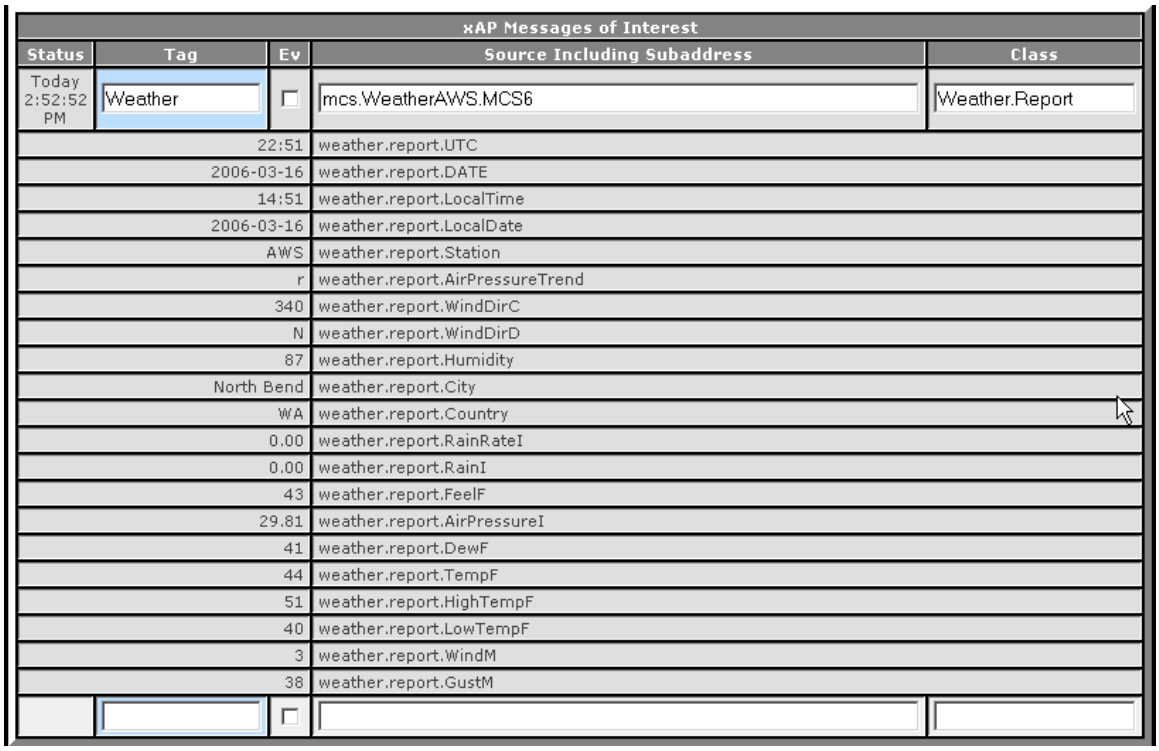
The “Tag” and “Ev” fields are associated with scripting. The Tag value entred will be the name used when referencing the valve from a script. The Ev (event notification) checkbox is used if the script should be called when the status of the value changed. When the script is called the tag name is passed as an input parameter to the script. The actual scripting interface is described in section 11.

## 10.4 xAP Messages of Interest

The scripting interface supports use of received xAP messages. Four controls are provided. The first two, “Tag” and “Ev” operate in the same manner as these same fields in section 10.3. The Tag is the name used to reference the message and Ev is used to trigger the script when the message is received. When the script is called the tag name is passed as an input parameter to the script. The actual scripting interface is described in section 11.

The address, including subaddress, of the message and the class of the message are also entred to identify the message for which data will be collected for scripting use.

The Receive xAP Detail shows the Key/Values of the message. These are always available via scripting even when not displayed on the Browser.



xAP Messages of Interest

| Status | Tag | Ev | Source Including Subaddress | Class |
|---|---|---|---|---|
| Today 2:52:52 PM | Weather | ☐ | mcs.WeatherAWS.MCS6 | Weather.Report |
| 22:51 | | | weather.report.UTC | |
| 2006-03-16 | | | weather.report.DATE | |
| 14:51 | | | weather.report.LocalTime | |
| 2006-03-16 | | | weather.report.LocalDate | |
| AWS | | | weather.report.Station | |
| r | | | weather.report.AirPressureTrend | |
| 340 | | | weather.report.WindDirC | |
| N | | | weather.report.WindDirD | |
| 87 | | | weather.report.Humidity | |
| North Bend | | | weather.report.City | |
| WA | | | weather.report.Country | |
| 0.00 | | | weather.report.RainRateI | |
| 0.00 | | | weather.report.RainI | |
| 43 | | | weather.report.FeelF | |
| 29.81 | | | weather.report.AirPressureI | |
| 41 | | | weather.report.DewF | |
| 44 | | | weather.report.TempF | |
| 51 | | | weather.report.HighTempF | |
| 40 | | | weather.report.LowTempF | |
| 3 | | | weather.report.WindM | |
| 38 | | | weather.report.GustM | |
| | | ☐ | | |

**Figure 3 xAP Scripting Messages Of Interest**

### *10.5 Refresh*

Two control buttons are provided to simplity integration of xapmcsSonar as shown in **Error! Reference source not found.**. The "BSC Query of Receiver" button is used to have xapmcsSonar deliver xapbsc.info message for all the used devices. This is useful when integrating with something like Homeseer and devices discovery is needed to create the Homeseer devices.
The "Read Script xapmcsSonare.txt" button is used to reread the file \Script\xapmcsSonar.txt. This is the script file that is executed based upon periodic or triggered events. This allows the file to be edited and when edits are complete the updated version can then be applied without needing to restart xapmcsSonar.

## 11 Scripting Interface

A Tag is a user-defined name that is available from the scripting interface. The contents of the Tag name will be the current status of the component. This status can be obtained from script with the object's Request method.

For the Sonar device the name of the Tag, and one of the two special words "Value" or "LastChange" are used. For example:

```
Status = xap.Request("Sonar")("value")
LastChange = xap.Request("Sonar")("LastChange")
```

For xAP received messages the Key from the message body is used rather than “Value”. “LastChange” is also available and will return the time the messsage was received. For control of the sonar parameter from script using the object’s Response method with the following keys. Review the script example for proper syntax. Note that these scripted changes to the sonar nodes operating properties do not change the setup configuration that is done via browser and they only are active as long as the node remains running and not changed from the browser UI.

```
'Power = ON,OFF
'Ranging = 0 ..
'Hysteresis = 0 ..255
'Min = 0 .. 255
'Max = 0 .. 255
'Filter = 0 .. 99
'Sample = 0,1,2
```

All scripting performed locally under activation by xapmcsSonar is routed through xapmcsSonar.txt which is located in the \Scripts folder under xapmcsSonar.exe. This file is called periodically and on event based upon configuration settings in Section 10.

This local scripting is independent from other scripting such is available via Homeseer. It allows local control decisions to be made based upon sonar distance information and information available from xAP messages.

## *11.1 Methods*

### 11.1.1 xAP Message Related Methods

Public Sub Response(sTag As Variant, sValue As Variant)
    Command a Sonar property
Public Function Request(sTag As Variant) As Variant
    Get value from a key within a received message or average sonar measurement
Public Function TagList(sTag As Variant) As Variant
    Obtain collection of Tags
Public Function Sequence(sTag As Variant) As Variant
    Get relative number of last change of xAP Message
Public Sub X10(sDevice As Variant, sCommand As Variant, Optional sBrightness As Variant = 0)
    Send command using xap-x10 schema
Public Sub SendXapMessage(sTarget As Variant, sClass As Variant, sSection As Variant, sData As Variant, sSubaddress As Variant, sUIDIn As Variant, sTargetSubaddress As Variant)
    Send a general case xAP message

### 11.1.2 General Scripting Support

Public Sub Run(sFilename As Variant)

Public Function RunEx(sFilename As Variant, sProcedure As Variant, sParameter As Variant) As Variant
Public Function Spawn(sScriptName As Variant, sProcedure As Variant, sScriptParm As Variant) As Object 'WshExec
Public Function Sunset() As Variant
Public Function Sunrise() As Variant
Public Property Get GetAppPath() As String
Public Sub WriteLog(c As Variant, message As Variant)

### 11.1.3 Ini File Support

Public Sub SaveIniSetting(Group As String, KEY As String, Value As String, Optional sFile As String)
Public Function GetIniSetting(Group As String, KEY As String, Default As String, Optional sFile As String) As String
Public Function GetINISection(Group As String, Optional sFile As String) As String
Public Sub ClearINISection(Group As String, sFile As String)
Public Sub FlushIniCache()

## *11.2 Script Example – \Scripts\xapmcsSonar.txt*

```
sub main(sTag)

	'=================== Script Objective ============================
	'Send a Message.Display and Speak when Sonar reading over 10 inches
	 'Change the properties of Sonar node based upon ON/OFF status of Watt node
	'=================================================================

	'xap.writelog "Sonar Script",cstr(sTag)


	'When sTag is null it means that this is periodic call to the script
	'rather than on a data change event.  For this script there is nothing
	'to do on periodic called if it occurs

	if sTag = "" then
		exit sub
	end if


	'================ commentary on debug and getting message data ==========

	'write to the log for debug
	 xap.writelog sTag,"Sonar=" & xap.Request("Sonar")("value")
	 xap.writelog sTag,"Watt=" & xap.Request("Watt")("input.state.text")

	'Get the message info that caused this script to be called.

	'When using X10 or other schema that does not resolve to a unique address
	'in the header such as xapbsc does then "LastEvent" should be used
	'If "LastEvent" is not used then the values returned from the requested
	'collection will be the values from the last message requested for the
	'tagged item and it may not be the values from the message that generated
	'the event call.  e.g. F4 message follwed by F3 message will generate
	'and F4 event, but the F3 data may now be reflected in the current X10 tag
	'=============================================================================
```

```
	'==============================
	'Test Processing
	'==============================
	if sTag = "Sonar" then
		'native properties for internal items are "value" and "lastchange"
		'properties for received xAP messages will be a function of the schema

		status =  xap.Request("Sonar")("value")      'internal value example to
get average distance
		changed = xap.Request("Sonar")("lastchange") 'internal example to get time
of last change
		xap.writelog "Status/Changed", cstr(status) & "," & cstr(changed)

		if cint(xap.Request("Watt")("value")) > 10  then
			xap.SendXapMessage "","Message.Display","Display.Text","Line1" &
chr(1) & "Car on Road " & cstr(now),"","",""
			xap.SendXapMessage "","xap.Voice","Voice.Speak","Say" & chr(1) &
"Car on Road","","",""
		end if

	End if


	'====================================================================
	'Reference Example processing of an xAPBSC message using the State key
	 'and changing some of the properties of the Sonar processing
	'====================================================================
	if sTag = "Watt" then
		Value = ucase(xap.Request("Watt")("input.state.state"))
		if Value = "OFF" then
			xap.Response "Sonar","Power=Off"
		Else
			xap.Response "Sonar","Power=On"
			xap.Response "Sonar","Filter=75"
			xap.Response "Sonar","Ranging=20000"
			xap.Response "Sonar","Hysteresis=5"
			xap.Response "Sonar","Max=30"
			xap.Response "Sonar","Min=10"
			xap.Response "Sonar","Sample=1"
		End if
		exit sub
	End if


End Sub
```