MonitorThis

Contents

Overview

MonitorThis is an efficient, effective monitoring service.

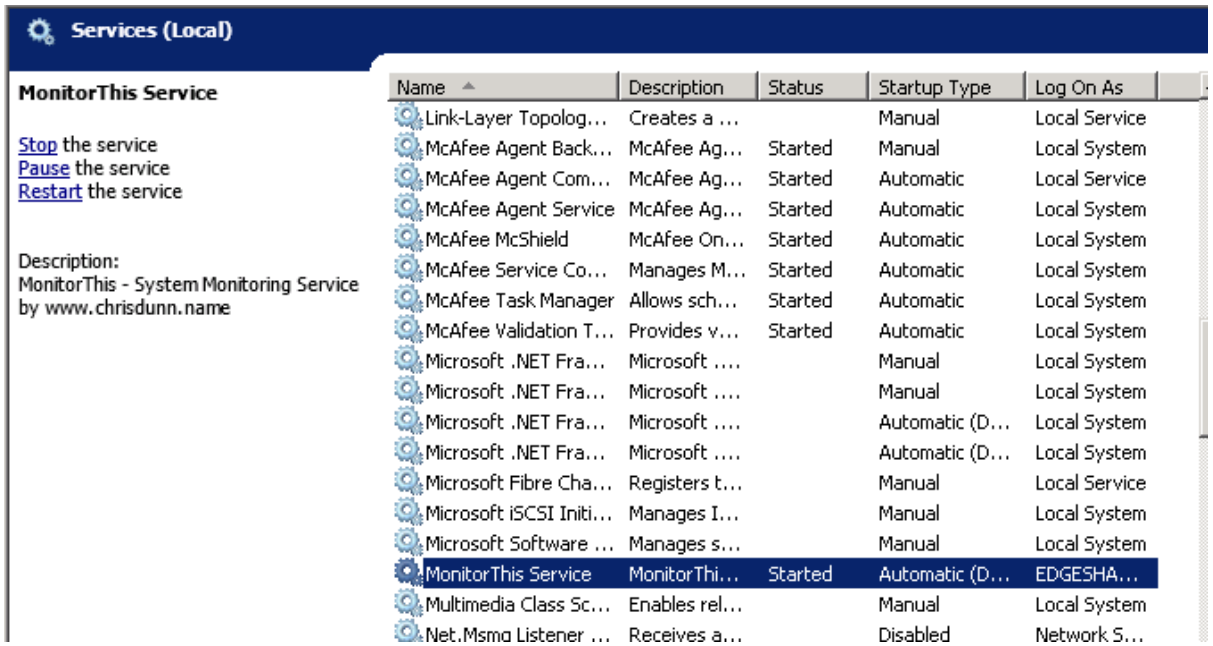
MonitorThis.exe

MonitorThis.exe is a single executable that is used to run, install, uninstall or update the service. Microsoft .Net 4.0 is required. There is no GUI, this is a command line only application.

/install – Install the service (installs from the startup location to the standard install location C:\ProgramData\MonitorThis) and copies across a matching configuration file if found, registers the service with Local System account and starts it.
/uninstall – de-registers the service (does not remove any files)
/update – Stops the service, copies the latest executable and configuration file for the current machine from the location the executable is run from to the standard install location (C:\ProgramData\MonitorThis) and restarts the service.

If no XML file is defined for the current machine a new file is created when the service starts. The service will install under the Local System account by default. It is recommended to change this to a unique account with access to the required resources.

Once installed MonitorThis can be found in Windows Services.



cache.xml

To ensure that each monitor runs to schedule even if the service is stopped and restarted MonitorThis uses a cache file automatically created in the local install folder (C:\ProgramData\MonitorThis). This cache file contains one record for each monitor name with the last run time of that monitor. This cache file is loaded when the service is started and changes made will not be reflected in monitoring actions until the service is restarted.

Deployment

The recommended approach for deployment is to store the MonitorThis.exe file and all XML configuration files in a network share accessible by the machines to be monitored. Running *MonitorThis.exe /install* from this location on a target machine will install the service and configuration file and start monitoring. Running *MonitorThis.exe /update* from this location will force an immediate update with the service being stopped, the latest files installed and the service being restarted. The *Update* monitor can be used to automatically check and update the application on each machine.

Configuration File

An XML file is used for configuration of all monitors on a machine. The XML file is stored in the install folder (C:\ProgramData\MonitorThis) and is named after the server (e.g. PC01.xml). This configuration is loaded when the service is started and changes made will not be reflected in monitoring actions until the service is restarted.

Base XML Format

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceConfig xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ServiceName="MonitorThis">
  <Monitors>
```

```
   <Monitor MonitorName="CPU" MonitorType="CPU" Enable="true" Schedule="Minutes"
Interval="1" Threshold="95" Alert="true" Email="false" />
</Monitors>
 <Email>
        <Server>10.2.0.71</Server>
        <Port>25</Port>
        <From> administrator@mail.com </From>
        <To> recipient@mail.com </To>
        <User></User>
        <Password></Password>
        <SSL></SSL>
        <URL></URL>
 </Email>
 <TimeCheckCycle>10000</TimeCheckCycle>
 <TimeStartDelay>2000</TimeStartDelay>
 <TimeStopTimeout>5000</TimeStopTimeout>
 <Logging>false</Logging>
 <Alerts>
        <Path>\\server\alerts</Path>
        <URL></URL>
</Alerts>
<Sleep>
        <Start>1000</Start>
        <Finish>0200</Finish>
</Sleep>
 <Description>My Application Server</Description>
</ServiceConfig>
```

An XML file is used for all monitor configuration on each machine. The XML file is stored in the install folder (C:\ProgramData\MonitorThis) and is named after the server (e.g. PC01.xml).

Service Parameters

```
<TimeCheckCycle>10000</TimeCheckCycle>
<TimeStartDelay>2000</TimeStartDelay>
<TimeStopTimeout>5000</TimeStopTimeout>
```

These parameters control the start and stop delays for the service and also how frequently it polls for actions to perform. These rarely need modification. All times are specified in milliseconds.

Description

```
<Description>My Application Server</Description>
```

This allows you to specify a detailed description for the server to be used in all email notifications.

Alerts

```
<Alerts>
        <Path>\\server\alerts</Path>
        <URL></URL>
</Alerts>
```

This specifies the location for alert HTML files to be saved. If the Path and URL are blank the path will default to the application install folder. Alert files are created when a problem is identified and the monitor is set to alert. If no problem is found on the next run the alert file is removed. Alert files are named **<server>-<monitor name>.html**.

Alert Files are created in Unicode to ensure compatibility with multiple languages. Alert files are HTML files which contain a header row and detail rows for alerts that can be easily integrated into other systems or a monitoring portal.

If a URL is specified then alert data is posted to the specified URL. This can be used in conjunction with file based alerts. See the Monitoring Portal section for more information.

Logging

<Logging>false</Logging>

If enabled a log file is generated in the application install folder. This should be set to false in production and is recommended for debugging only as it will generate a very large log file.

Email

```
<Email>
        <Server>8.8.8.8</Server>
        <Port>25</Port>
        <From>administrator@mail.com</From>
        <To>recipient@mail.com</To>
        <User></User>
        <Password></Password>
        <SSL>true</SSL>
        <URL></URL>
</Email>
```

This is the configuration for mail sending. User/Password/SSL are optional. If a URL is specified then email data is posted to the specified URL instead of being emailed directly from the local machine. See the Monitoring Portal section for more information.

Sleep

```
<Sleep>
        <Start>1000</Start>
        <Finish>0200</Finish>
</Sleep>
```

This is the configuration for adding a quiet period to monitoring. The start and finish time must be formatted in 24 hour HHMM notation as shown above. During this period all monitors will be disabled. As soon as the window has finished then monitoring will resume. Any monitors missed during this period will run after the window has finished but monitors will not be run more than once.

Monitors

Common Monitor Features

<Monitor MonitorName="CPU" MonitorType="CPU" Enable="true" Schedule="Minutes" Interval="1" Threshold="95" Alert="true" Email="false" />

This is an example of a standard monitor. Each monitor has the following attributes. The same MonitorType can be used in different Monitors within the same configuration file.
MonitorName – Unique identifier for the monitor (string)
MonitorType – The type of monitoring/action to occur (enumeration)

- CPU
- PhysicalMemory
- VirtualMemory
- DiskSpace
- DNS
- Process
- Service
- Update
- Execute
- FileExist
- WebRequest
- MaxChildren

Enable – Activates/Deactivates the monitor (Boolean)

- True
- False

Schedule – Frequency/Trigger for this monitor (enumeration)

- Days
- Hours
- Minutes
- Seconds
- Time

Interval – This is the value to be applied to the schedule above to specify the frequency/trigger for the monitor. For "Time" the interval is the time in 24 hour notation i.e. 0800 or 1400. (integer)
Threshold – This is used for setting monitoring/alert threshold levels (vary by monitor) (double)
Alert – This specifies whether alert files should be created when a monitor identifies a concern (Boolean)

- True
- False

Email – This specifies whether an email should be sent when a monitor identifies a concern (Boolean)

- True
- False

CPU

<Monitor MonitorName="CPU" MonitorType="CPU" Enable="true" Schedule="Minutes" Interval="1" Threshold="95" Alert="true" Email="false" />

This monitor measures CPU usage when it runs. It takes an average of two readings 2 seconds apart. The threshold is the %CPU used. An alert or email is triggered when this level is met or exceeded.

Physical Memory

<Monitor MonitorName="PhysicalMemory" MonitorType="PhysicalMemory" Enable="true" Schedule="Minutes" Interval="5" Threshold="95" Alert="true" Email="false" />

This monitor measures Physical Memory usage when it runs. The threshold is the %Memory used. An alert or email is triggered when this level is met or exceeded.

Virtual Memory

<Monitor MonitorName="VirtualMemory" MonitorType="VirtualMemory" Enable="true" Schedule="Minutes" Interval="5" Threshold="95" Alert="true" Email="false" />

This monitor measures Virtual Memory usage when it runs. The threshold is the %Memory used. An alert or email is triggered when this level is met or exceeded.

Disk Space

<Monitor MonitorName="DiskSpace" MonitorType="DiskSpace" Enable="true" Schedule="Minutes" Interval="5" Threshold="95" Alert="true" Email="false" />

This monitor measures Disk Space usage on all drives when it runs. The threshold is the %HDD used. An alert or email is triggered when this level is met or exceeded.

DNS

```
<Monitor MonitorName="DNS" MonitorType="DNS" Enable="true" Schedule="Minutes" Interval="5" Threshold="0" Alert="true" Email="false">
    <Hosts>
        <string>google.com.au</string>
    </Hosts>
</Monitor>
```

This monitor checks that DNS can resolve host names. For each host name specified as a string this monitor attempts to resolve the name to an IP. Threshold is not used. An alert or email is triggered if the host cannot be resolved.

Ping

```
<Monitor MonitorName="My Pings" MonitorType="Ping" Enable="true" Schedule="Minutes" Interval="5" Threshold="60" Alert="true" Email="false">
    <Hosts>
        <string>www.microsoft.com</string>
    </Hosts>
</Monitor>
```

This monitor pings the specified hosts and checks for a successful response within the time period (threshold) specified. The Threshold is the timeout period in seconds. An alert or email is triggered if a successful ping is not received in this time.

Process

```
<Monitor MonitorName="Systems" MonitorType="Process" Enable="true"
Schedule="Minutes" Interval="5" Threshold="3" Alert="true" Email="true">
        <Processes>
                <string>SQL Server%-sSQLEXPRESS%</string>
                <string>svchost.exe%LocalService</string>
                <string>SearchIndexer%</string>
        </Processes>
</Monitor>
```

This monitor checks individual running processes. For each process specified as a string this monitor checks if a single process with the specified command line is running. The search uses an 'ends with' function to check if there is only 1 process which contains the string specified in its command line. The % character can be used as a wildcard as shown above to find processes containing multiple sequential strings or at the end of the string to make it a 'contains'. You can identify the command line for windows processes using Task Manager. The threshold specified is the maximum memory use for any of the processes specified in GB. An alert or email is triggered if the search for any matching process does not find a result, finds more than 1 result or the found process matches or exceeds the memory threshold specified.

Services

```
<Monitor MonitorName="Services" MonitorType="Service" Enable="true"
Schedule="Minutes" Interval="5" Threshold="0" Alert="true" Email="true">
        <Services>
                <Service ID="wuauserv" Start="false" />
                <Service ID="W3SVC" Start="true" />
        </Services>
</Monitor>
```

This monitor checks individual running services. For each service specified by the service ID above this monitor will check if that service is running. The ID of each service is the Service Name specified in the Services manager or Task Manager. If it is not running and Start is true then the service will be started. An alert or email will be triggered if a specified service is not found or not running.

Update

```
<Monitor MonitorName="Updates" MonitorType="Update" Enable="true"
Schedule="Minutes" Interval="15" Threshold="0" Alert="false" Email="false">
        <Folders>
                <string>\\myserver\updatepath</string>
        </Folders>
</Monitor>
```

This monitor is designed for automatic updating of the MonitorThis application and configuration file. This monitor will check each update path specified as a string to determine if the MonitorThis executable and a configuration file for the current machine exist. If they do and they are newer than

the current version then the files are copied to a 'temp' folder below the install location and then an update is triggered which will shut down the service, update the files and start the service. Threshold is not used. An alert or email will be triggered if the update process does not complete successfully.

Execute

```
<Monitor MonitorName="Start Programs" MonitorType="Execute" Enable="true"
Schedule="Minutes" Interval="5" Threshold="0" Alert="true" Email="false">
    <Commands>
        <string>c:\windows\system32\cscript.exe
        "\\myserver\myfolder\myvbscript.vbs"</string>
        <string>C:\Windows\System32\cmd.exe /C
        C:\myprogram\mybatchfile.cmd</string>
        <string>"C:\Program Files (x86)\Microsoft
        Office\Office14\Winword.exe"</string>
    </Commands>
</Monitor>
```

This monitor starts any windows applications, scripts or commands. For each command specified as a string the application will launch an individual process. The first part of each string must be a fully qualified locally stored application. Additionally you can add any parameters. The examples above show how to launch a program, network visual basic script and a batch file. The command launched will run non-interactive and will not be visible to any users so should be able to run without a user interface. An alert or email is triggered if the application fails to execute the specified command. Commands are executed in order. The threshold can be used to set the maximum wait in minutes before starting the next command. If threshold equals 0 then the application will wait for the previous command to completed before running the next. If the Threshold is equal to 1 if the previous command has not completed after 1 minute the next command will start anyway.

FileExist

```
<Monitor MonitorName="Output Files" MonitorType="FileExist" Enable="true" Schedule="Hours"
Interval="1" Threshold="-60" Alert="true" Email="true">
    <Folders>
        <string>d:\output\myfiles\</string>
    </Folders>
    <Files>
    </Files>
</Monitor>
```

This monitor checks if files exist. For each file or folder specified as a string the monitor will check if the file (or any files in the folder) exist. Both the Folders and Files sections must exist even if only one is used as shown above. The threshold can be used to specify a timeframe in minutes with a negative sign used to indicate direction. A threshold of 0 will check if any files exist. A threshold of 60 will check if files exist that were updated in the last 60 minutes. A threshold of -60 will check if files exist that were updated prior to the last 60 minutes. An alert or email is triggered if the file or files exist that meet the configured threshold.

WebRequest

```
<Monitor MonitorName="Website Check" MonitorType="WebRequest" Enable="true"
Schedule="Minutes" Interval="1" Threshold="0" Alert="false" Email="false">
```

```
        <URLs>
            <string>http://www.chrisdunn.name</string>
        </URLs>
    </Monitor>
```

This monitor checks that a successful response is received from any given URL. A web request is performed to each URL specified as a string. Any specified redirects will be automatically followed to the final destination. An alert or email is triggered if no successful response is received within 30 seconds. Threshold is not used. Using the special parameters {COMPUTER} or {DOMAIN} in a URL string will have these values automatically replaced at runtime with the actual values for the current machine.

MaxChildren

```
<Monitor MonitorName="ActiveProcesses" MonitorType="MaxChildren" Enable="true"
Schedule="Minutes" Interval="1" Threshold="10" Alert="true" Email="false"/>
```

This monitor checks the number of child processes launched by MonitorThis. A count of child processes is performed where the MonitorThis process ID is the Parent Process ID. This can be used to check if there are problems with batch files, vbscripts or other third party applications launching but not completing. An alert or email is triggered if the number of child processes is greater than the Threshold specified.

EventLog

```
<Monitor MonitorName="Events" MonitorType="EventLog" Enable="true"
Schedule="Minutes" Interval="5" Threshold="2" Alert="true" Email="true">
        <Events>
            <Event Log="Application" ID="1000" Type="" Source="" Message="w3wp" />
            <Event Log="System" ID="" Type="Error" Source="" Message="COM" />
        </Events>
</Monitor>
```

This monitor checks the Windows Event logs. Each scheduled run it will use the criteria specified to identify any corresponding events since the last check and generate an alert if events are found exceeding the threshold specified. Criteria with no value are excluded from the search. The Message criteria searches the contents of the event log message for the words specified. Types include Error, Warning, Information, Security Audit Success and Security Audit Failure.

Monitoring Portal

MonitorThis is designed to allow integration with a custom portal to manage and highlight alerts. To assist a Classic ASP portal is provided with this application. This portal can be monitored to suit your individual requirements. This portal allows some features to be handled by Web Services and not directly by the Monitors. To validate the status of monitoring a WebRequest monitor can be configured to send the machine name to a status page (status.asp) at a regular interval. In the example pages provided the current time is saved to a unique file for each machine. To visualise the activity of each machine the portal can then summarise all alert files in a specified folder for each machine and to indicate status (default.asp).
Alerts that run infrequently can be cleared from the website manually after being actioned (clearalerts.asp) and extensive detail of current activity and alerts on any server can be viewed directly from the portal (detail.asp).

The monitoring portal is designed to read Unicode and non-Unicode alerts allowing custom scripts to be used to complement the inbuilt monitoring features included in MonitorThis. Email alerts and file based alerts can also be posted from the application to the portal (alerts.asp and emails.asp) instead of being saved directly to a network share or sent directly allowing the portal to cater for WANs.

Default.asp (Portal Main Page)

This page displays all active alerts.

Detail.asp

This page displays details for any specified machine from a GET request and is triggered from the default portal page. The request to this page only has one parameter.

Parameters:

- COMPUTERNAME – Name of Machine

Clearalerts.asp

This page clears all alerts from a specified server from a GET request and is triggered from the default portal page. The request to this page only has one parameter.

Parameters:

- COMPUTERNAME – Name of Machine

Status.asp

This page can receive a web request from individual machines (using a GET request from the WebRequest Monitor) and creates an indicator file in the STATUS folder which the portal uses to provide a monitoring status indicator. This request only has one parameter.

Sample Monitor Configuration:

```
<Monitor MonitorName="Status" MonitorType="WebRequest" Enable="true" Schedule="Minutes" Interval="1" Threshold="0" Alert="false" Email="false">
  <URLs>
   <string>http://server/portal/status.asp?COMPUTERNAME={COMPUTER}</string>
  </URLs>
 </Monitor>
```

Parameters:

- COMPUTERNAME – Name of Machine

Alerts.asp

This page creates alert files in the ALERTS folder upon receipt of a GET or POST request from individual machines. These are then used to display the alerts on the portal. The URL should be specified in the configuration file.

Sample XML Configuration:

```
<Alerts>
        <Path></Path>
        <URL>http://server/portal/alerts.asp</URL>
</Alerts>
```

Parameters:

- COMPUTERNAME – Name of Machine
- MONITOR – Name of Monitor
- ALERT – Name of Alert

Emails.asp

This page sends emails upon receipt of a GET or POST request from individual machines. This allows emails to be sent from only one machine in the network (i.e. if only the portal server has email relay permission) but submitted from each individual machine and with different configuration. The minimum information required is SERVER, FROM, TO. The URL should be specified in the configuration file.

Sample XML Configuration:

```
<Email>
        <Server>10.2.0.71</Server>
        <Port>25</Port>
        <From> administrator@mail.com </From>
        <To> recipient@mail.com </To>
        <User></User>
        <Password></Password>
        <SSL></SSL>
        <URL>http://server/portal/emails.asp</URL>
 </Email>
```

Parameters:

- FROM – From email address
- TO – To email address
- CC – cc email address
- BCC – bcc email address
- SUBJECT – email subject
- HTMLBODY – html body content
- TEXTBODY – text body content
- SERVER – smtp server
- PORT – smtp port (default = 25)
- USER – smtp username
- PASSWORD – smtp password
- SSL – use SSL (default = false)

PostToForm.vbs

This is a sample vbscript that demonstrates how to post to the portal site from a vbscript. This code can be integrated into existing vbscripts or modified to send alerts from other programs or batch files allowing them to be integrated with the portal emailing or alerts functionality.

Sample Code:

```
HTTPString = ""

Function AddToHTTPString(variable,value)
	If len(HTTPString)=0 then
		HTTPString = variable & "=" & ReplaceVar(value)
	Else
		HTTPString = HTTPString & "&" & variable & "=" & ReplaceVar(value)
	End If
End Function

Function ReplaceVar(HTML)
	arrBefore = Array("%"," ","!","*","(",")",";",":","=","+","$",",","/","?","#","[","]")
	arrAfter =
	Array("%25","%20","%21","%2A","%28","%29","%3B","%3A","%3D","%2B","%24","%2C","%2
	F","%3F","%23","%5B","%5D")
	for i = 0 to uBound(arrBefore)
		HTML = Replace(HTML,arrBefore(i),arrAfter(i))
	next
	ReplaceVar=HTML
End Function

Function HTTPPost(sUrl, sRequest)
	Set encoding = CreateObject("System.Text.UTF8Encoding")
	sRequest = encoding.GetBytes_4(sRequest)
	Set encoding = Nothing
	set oHTTP = CreateObject("Microsoft.XMLHTTP")
	oHTTP.open "POST", sUrl,false
	oHTTP.setRequestHeader "Content-Type", "application/x-www-form-urlencoded"
	oHTTP.setRequestHeader "Content-Length", Len(sRequest)
	oHTTP.send sRequest
	HTTPPost = oHTTP.responseText
End Function

AddToHTTPString "COMPUTERNAME","SERVER"
AddToHTTPString "MONITOR","Test"
AddToHTTPString "ALERT","<tr style='height:14.25pt;Background:#BFBFBF;'><td>Sample
Alert</td></tr><tr><td>This is an alert for <u><strong>Test</strong></u> on SERVER.</td></tr>"

HTTPPost "http://localhost/portal/alerts.asp",HTTPString
```