



# Taking action with *Sentry-go Monitoring Automatic Responses*

Last Updated Thursday, 19 April 2012

© 3Ds (UK) Limited  
<http://www.Sentry-go.com>

*Be Proactive, Not Reactive!*

---

## Introduction

A typical monitoring solution will monitor your software or environment and, in the event an issue is detected, let you know. It may allow you to be alerted in a number of ways, depending on the fault and even the day or time the fault occurs. Some, like the Sentry-go Monitoring System go further by giving you the option of allowing the monitor itself to attempt to resolve the error itself by performing an automated action or response. This can be an extremely powerful feature and is certainly a subject worthy of more than a quick glance.

In the pages that follow, we'll explore the potential of this feature and explain some of the "dos and don'ts" when it comes to configuring these options so you can see for yourself, how it might help your organisation.

## The Benefits

There are a number of reasons why you might consider configuring automatic responses ...

- Some faults can actually occur quite frequently. They're more annoying than anything else, but require time and effort to resolve, *each & every time they occur!*
- Even with the quickest forms of alert, a manual response will be delayed. You may have to access a machine & logon, you may also be involved with another issue, or simply be on the phone and not see the message for a few minutes.
- For important issues or systems, every second counts. The faster the problem is resolved, the less people (or other systems) are affected - which is ultimately better for them, you and your business.

Of course, we're not saying everything can, or should be automated and we're certainly not saying the Administrators shouldn't be kept informed. But take the following examples ...

- A key service stops running on your server
- Your server runs low on disk space preventing prints from being submitted or data being written to the database
- Someone runs Setup on your live server without the Administrators knowledge
- A database table becomes locked such that your orders cannot be accepted
- Your web server returns error pages rather than your shiny home page

In all of these, and many more, you could easily take advantage of an automated approach, at the very least to get the system back & running in the quickest possible time, and investigate further thereafter.

For example, you might want the system to ...

- Restart the failed service
- Clear down temporary files to try and reclaim free disk space
- Check for a blocking SQL transaction & even terminate it if it's been holding locks for too long

### **Example**

Let's look at the last example in more detail. You have live SQL Server databases running one of your main business applications which accepts orders over the web. Your application therefore runs pretty much 24 hours a day. The web application interacts with SQL Server, as do other systems, some of which run overnight – extracts for nightly reports, housekeeping etc.

Now let's suppose an issue occurs in one of the other applications such that it locks the orders table or row. The application is waiting for other communication & hence waits. But during this time, on-line orders are also being received. The orders table is locked and hence the new data can't be saved; it's now 2am and orders from around the world are pending for your on-line users.

Actually of course, they're blocked, waiting for the other system to continue. This blocking may ultimately lead you to question the system design & look to rectify it long term, but in the short term, it's arguably the orders – from actual or potential customers that need to be received.

What would you do manually ? As the DBA or Administrator you'd probably take a look at the database, processes & locks and identify which process is causing the problem. You might then note down the details – what it last did etc. through DBCC or SQL Server Management tools, then terminate the offending (blocking) process.

If it's 2am this will take time, assuming you get to find out about it all until the following morning. A better approach might be to automate those exact tasks ...

- The monitor detects the blocking & verifies how long the blocks have been held
- If the allowed timeframe has been exceeded it ...
  - Logs details of the offending SQL processes (users, SQL statements etc.)
  - Terminates the blocking process
  - Alerts the Administrator if required
  - Monitors to ensure no further locking is evident

With this approach, web users would probably not even notice an issue, just a small delay in placing their order, so you can see just how powerful a well thought out strategy can be.

For situations like these, it makes sense to let the automation do it for you, so that you can concentrate on the longer term implications of the issue and hopefully keep one step ahead of the problem.

## Consider your Actions

When configuring auto responses, the first & most obvious thing to consider is what action, if any, the monitor should take. Sometimes, this will be obvious – a command or specific action. On other occasions, maybe you simply want to be informed of the error & don't want anything else to happen. For example, in these situations ...

- When the machine runs low on memory
- When the runs more than the expected number of processes
- A higher than expected number of logon attempts fail

... there's not really anything specific to do except investigate why they occurred and their longer term resolution. If however, you can & do choose to configure an automatic response, you can select a number of different types.

### Run a Pre-configured Response

Sentry-go monitoring components come pre-configured with a number of actions, relating to the task being monitored. They're designed to make life easy and often you simply need to select the appropriate action in the "Response" tab and let the monitor do the rest.

Examples include ...

- Restarting or continuing a stopped or paused service.
- Terminate a blocking SQL query.
- Pausing a queued document that is considered too large for the printer.

## Run an “Implicit” Script

“Implicit” scripts are selectable responses that actually cause a script to be generated to perform the required functions. The Sentry-go Scripting Wizard is automatically invoked in this case, allowing the file to be generated without the need for coding or script knowledge.

[Click here for more information on the Scripting Wizard.](#)

## Run a Windows command

Alternatively, there may be a command you would use on the command line as a first resolution. These are direct commands you might run on the command line having logged on to the server. For example ...

- Net start <ServiceName>.
- Net use <Mapped Drive Information>

## Write & run a batch file containing one or more commands

Batch files allow a group one or more Windows commands or applications to be grouped & run in sequence. Basic logic can also be performed within the batch file.

## Write & run a script (e.g. VB Script) containing logic to perform the required response

This is often considered one step up from the batch file and allows far greater programmatic control and access. If you're a developer, or know scripting such as VBScript, then you can perform a vast array of functions and access even more through extendible objects such as Microsoft's CDO (for e-mail), the File System Object (FSO for files) and ADO for database access.

Complex, parameterised & conditional logic can be performed using scripting.

## Reboot the local (monitored) server

Often a last resort, there may be occasions when the primary role of the server has been compromised to such an extent that the first thing you'd do is restart Windows. A system reboot asks Sentry-go to restart the local server – the server on which the monitor is running.

To guard against a never ending loop, in-built logic within Sentry-go helps to ensure that the same check failure doesn't cause a system reboot to be repeated.

## Consider All the Options

Although the automatic response options given above are designed to be easy to use and quick to configure, it's worth considering all the options. Sometimes, for example, you may want to try one thing and if that fails, do something else. For these scenarios, the script is an obvious candidate, even if one of the actions can easily be performed using a different technique.

For example, if a specific service fails, you might want to restart other related services at the same time, checking their status as you go. A simple script could easily be written to stop, then start each in turn (or stop all, then restart them in sequence).

## Some Dos & Don'ts

### **Make sure all paths are local, or at least accessible from the monitoring server.**

It's the Sentry-go monitoring service that runs the response & so all files must be accessible to it. This includes any files or commands contained inside a batch file or script that's being called.

Although the Sentry-go Client Console allows you to configure a system remotely, remember that paths must be local - or at the very least accessible to the local server over the network.

The use of local paths are recommended – see below.

### **Avoid mapped drives for network paths – use UNC names instead**

Always configure paths using a full UNC path - [\\Server\Share\Folder](#) etc. and avoid referring to a network drive using a mapped name.

For example, you may map the M: drive to a particular server but when referring to a file or folder on that server, use the full server & share name (in UNC format) and avoid using M: in the path.

Sentry-go performs its monitoring & auto-responses from a Windows service. Although this typically as local administrative permissions, it will often have no access to the mapped drives your user session has. Most mappings are created by Windows Explorer when you logon; Explorer doesn't run for a service and hence no mapping will be available.

### **Use fully qualified paths**

Within the file or script, ensure any files, paths or executables are qualified with the full path. This ensures that wherever the script is run from (its current directory), the path will still be valid.

## Consider the “current directory”

Some utilities expect to run from the current directory, when run from batch files. To ensure the correct path is set, "CD" into the appropriate directory before running the utility. For example ...

```
CD \new-path\new-folder  
MyCommand.exe
```

Also, note that if the target directory is (or could be) different from the drive on which the monitor is running you should change to the new drive before CD'ing into the target directory. In other words, don't attempt to change drive within the CD command. If you do, the command may not work as you expect. For example, to ensure you're running on the D: drive ...

```
D:  
CD \new-path\new-folder  
MyCommand.exe
```

## Access network resources carefully

If your response does rely on a network resource, configure your response carefully. There are a couple of reasons for taking care ...

- The network or resource may not be available when you attempt to run the response. If this happens, the response will fail, even though the actual solution would, if run, have resolved the problem.
- The user running the response must have access & permission to access the network resource.

By default, the user running the Sentry-go service will run a response & typically this will be the Local System account. Whilst great for local issues, system users have limited or no access remotely or on the domain & hence any access will fail.

To overcome this ...

- Configure the response to run as a domain user (with the appropriate access)
- Run the Sentry-go monitoring service as a domain user – with the appropriate permissions locally & on the domain.

## **Remember who's running the script (it's probably not you)**

When you run the script from a command prompt or Start/Run, you are implicitly running it under your own Windows user account.

When you run it as a response, however, by default it's the user running the monitoring service that's in charge. This can have an impact on the way it works (or doesn't work).

Specifically, your user ID is most likely to be a domain user with access to both local & network resources. By default, however, the monitoring user is the Local System user – which has Administrative privileges locally but few or no access to the domain or network. This may be important, depending on what the script is going to do.

If your script is likely to need domain access ...

- Run the response as a named (domain) user.
- Run the monitoring service as a domain user (with local administrative privileges).

## **Configuring the User ID to run the script**

If you can't, or don't wish to use the Local System account, it is recommended that a dedicated domain user account is configured for use by Sentry-go. This way, you can ...

- Configure it to run Windows services (required if running the Sentry-go monitoring service).
- Grant it the appropriate permissions required on the domain.
- Grant it Administrative privileges on the local server

## **Remember where the response is being run from**

The response is always run by the monitoring service – i.e. on the machine running the Sentry-go monitor that detected the fault. Even if remote UNC are used, it's the local server that's running the file.

It is strongly recommended that responses are run locally. However, if remote access/running is required, the script itself must be coded to access remote resources.

See below for more information.

## **Consider what your file or script will access**

In some cases, your script may rely on other technologies or installed applications to run (MAPI for example). This is normally fine, but if these technologies also rely on others, or they require a logged on user (or at least an available user profile) you will need to ensure the following ...

- Check that each component is installed on your server.
- Check that the user running the response has access to, and can run these components.

Also, some technologies assume a user is running them from a desktop window. With Sentry-go, there's no desktop window as the service itself is running the job and by default, no user profile will be available (as the local system user doesn't have an interactive session).

To overcome this, either configure the response to run as a domain user (this will load the user's domain profile), or run the Sentry-go monitoring service as a domain user – with the appropriate permissions locally & on the domain (which will also cause their profile to be loaded when the monitor is run).

### **Don't display an interface (GUI)**

Windows services (such as the monitoring service) typically run without a user interface or Windows session. In fact, most likely of all, no user will be logged on directly to the server.

With this in mind, you must avoid calling or running applications which contain a user interface (including message boxes etc.) as this window ...

- Will not be seen (even if you are logged on to the server)
- Cannot be accessed (or closed).

### **Ensure the script doesn't take too long to run**

After a response is initiated, Sentry-go will wait for it to complete before proceeding. It determines this by monitoring the job itself and, once it closes (and therefore exits), re-checks the monitored item.

During this time, the monitoring component calling the response will wait (as other checks may be affected by the result) and hence the quicker the response, the faster the monitor will be at performing other checks.

To protect itself against a hung response, Sentry-go, by default will only wait so long for the script or file to run. If it hasn't completed within this period, it will time out and the response deemed to have failed.

- Try to ensure the response works as quickly & efficiently as possible.
- If a longer response must be run, you can increase the default timeout, so that the monitor allows additional time (though this is not recommended as it could delay other monitoring tasks).

## Local vs Remote Responses

Sentry-go is designed to run on the server being monitored. There are a number of reasons for this, for example ...

- Local monitoring is much more efficient than performing it across the network
- There is no network overhead
- There is no reliance on the network. If a network link should fail, it will not impact on the monitor

For the same reasons, we strongly recommend that response files & scripts are located and run directly on the local server – i.e. the server where the monitoring service is running. This ensures that resources should be available & the monitor will have access to them.

### Running Remote Responses

Although the use of local responses is recommended, there may be times when it's more appropriate to perform a remote action. For example ...

- The event that generated the error was caused by an issue on a remote server.
- The monitor check itself was checking something remotely across the network – e.g. via a script.

In these cases, where corrective action may be needed elsewhere, you might consider using a remote response. How you achieve this will depend on the facilities available to you, the version of Windows being run and the access you have available from the monitoring machine.

Below are some examples of how this might be achieved. *Note that these are just examples designed to give you an idea of what's involved. They are designed to be used without further amendment & development.*

## Use Windows Management Instrumentation (WMI) Objects

Windows Management Instrumentation or WMI is an optional technology available to Windows. It provides a number of objects (classes) that allow you to interact with different parts of the software & hardware system. More importantly, it also allows you to do this on the local or remote machine.

For example, you could stop/start one or more Windows services - or even restart the Windows installation remotely, you can use a script containing WMI commands. WMI must be installed on the monitoring & target servers.

```
` Declarations
Dim strWQL
Dim objWMI
Dim colResult
Dim objWinService
Dim intResult
Dim strTargetServer
Dim strTargetService

` Error handling
On Error Resume Next

` Set target server name & service
strTargetSvr = "Your Server name"
strTargetSvc = "Your Service name"

` Access WMI
Set objWMI = GetObject("winmgmts:\\\" & strTargetSvr & "\root\cimv2")
If Err <> 0 Then

    ` Failed
Else

    ` Query WMI
    strWQL = "SELECT * FROM Win32_Service WHERE Name = '\" & strTargetSvc & '\""
    Set colResult = objWMI.ExecQuery(strWQL)

    ` Check results
    If colResult.Count > 0 Then

        For Each objSvc in colResult

            ` If we want to start it
            If objSvc.Started = False Then

                intResult = objSvc.StartService
                If intResult <> 0 Then

                    ` Failed
                    End If
                End If

            ` If we want to stop it
            If objSvc.Started = True Then

                intResult = objSvc.StopService
                If intResult <> 0 Then

                    ` Failed
                    End If
                End If
            End If
        Next
    End If

    ` Cleanup
    Set objWMI = Nothing
End If
```

## Use the WMI component to remotely execute a command file or script

Using the above WMI components, you can also connect to a remote machine and run one or more commands, files or scripts. Using this feature you could host batch files & scripts on the target server and call them from the monitoring machine as required.

To do this you would need to ...

- Connect to WMI on the remote server
- Access the Win32\_Process object of WMI
- Ensure the user running the script has permissions to access & run remote commands through WMI on the remote server
- Or specify user credentials within the script to connect to WMI with a given user ID before starting the process.

```
` Declarations
Dim objWBEMLocator
Dim objWMI
Dim objProcess
Dim objStartup
Dim objConfig
Dim strTargetSvr
Dim strTargetUser
Dim strTargetPassword
Dim strTargetCommand
Dim intResult

` Constants
Const cWBEM_IMPERSONATE = 3
Const cWBEM_PKT_PRIVACY = 6
Const cREMOTE_EXEC = -1

` Error handling
On Error Resume Next

` Set target details
strTargetSvr = "Your Server name"
strTargetSvc = "Your Service name"
strTargetUser = "Your user name"
strTargetPassword = "Your password name"
strTargetCommand = "The command to run (must be accessible to remote server)"

` Access WBEM
Set objWBEMLocator = CreateObject("WbemScripting.SWbemLocator")
If Err.Number <> 0 Then

    ' Failed

End If

` Connect to QMI
Set objWMI = objWBEMLocator.ConnectServer _
    (strTargetSvr,"root\cimv2", strTargetUser, strTargetPassword)
If Err.Number <> 0 Then

    ' Failed

End If

objWMI.Security_.ImpersonationLevel = cWBEM_IMPERSONATE
objWMI.Security_.AuthenticationLevel = cWBEM_PKT_PRIVACY
```

```

' Configure the new process
Set objStartup = objWMI.Get("Win32_ProcessStartup")
If Err.Number <> 0 Then

    ' Failed

End If

Set objConfig = objStartup.SpawnInstance_
If Err.Number <> 0 Then

    ' Failed

End If

objConfig.ShowWindow = SW_NORMAL

Set objProcess = objWMI.Get("Win32_Process")
If Err.Number <> 0 Then

    ' Failed

End If

intResult = Process.Create (strTargetCommand, NULL, NULL, intProcessID)
If Err.Number <> 0 Then

    ' Failed - result code indicates reason

End If

' Cleanup
Set objWMI = nothing
Set objWBEMLocator = nothing

```

### **Use 3<sup>rd</sup> party remote execution technology**

In addition to WMI, other remote execution technologies are available with Windows and through 3<sup>rd</sup> parties. For example, "RSH" in Windows XP. These can often be used from the command line, and therefore batch files & scripts to remotely execute logic remotely.

## Conclusion

As we have seen, automatic responses are a powerful feature of Sentry-go, allowing you to recover & resolve known issues without the sometimes lengthy delays of manual access. With careful planning, they can be of great benefit, and take advantage of other technologies that best fit your environment.

## More Information, Help & Support

More information can be found in the guides that accompany the Sentry-go software. You can also access the following resources ...

- For the very latest information & product updates, please visit <http://www.Sentry-go.com>
- For sales advice, please e-mail [Sales@Sentry-go.com](mailto:Sales@Sentry-go.com)
- For technical support, please e-mail [Support@Sentry-go.com](mailto:Support@Sentry-go.com)

