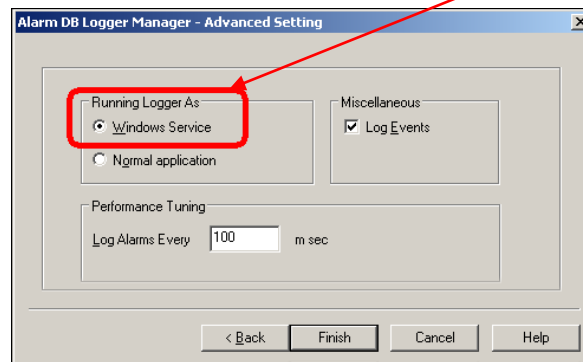

NOTE: This article and all content are provided on an "as is" basis without any warranties of any kind, whether express or implied, including, but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. In no event shall Wonderware North be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of information contained in this article.

Purpose:

The InTouch Alarm DB Logger utility is used to log alarms and events from InTouch or System Platform to a Microsoft SQL Server database. Once this information is saved to the database, you can use a set of database views to easily query past and current alarm and event occurrences for a variety of reports.

In many installations, an engineer would configure this alarm logger to start as a Windows Service, which would enable the logger to start automatically when the computer boots up.



The benefit of this type of configuration is that no user interaction is required to log the alarms and events to the database after the computer starts up.

However with Microsoft's Windows 2008 Server Operating System, there have been some security restrictions around services and their ability to interact with desktop applications. Due to that fact, the Alarm DB Logger Manager cannot be configured to run as a Windows Service. This TechTip can be used as a workaround for these new Operating System restrictions.

NOTE: The Alarm Logger instance generated by this object will log alarms and events generated by the System Platform objects only. If logging is required from stand-alone InTouch applications, a separate Alarm Logger instance will need to be run on a different node.



Procedure:

1. Create a new derived template from a UserDefined template (when using the Base Template Library, you should derive from the \$mUserDefined template) and name it **\$ProcessControl**.
2. Create seven (7) new UDA's (User Defined Attributes):

Name	Data Type	Category	Array	Initial Value
_Debug	Boolean	User Writeable	No	Off/False ¹
Process_Automate	Boolean	User Writeable	No	On/True ²
Process_FilePath	String	User Writeable	No	See Note ³
Process_Name	String	User Writeable	No	See Note ⁴
Process_Running	Boolean	Object Writeable	No	Off/False
Process_Start	Boolean	User Writeable	No	Off/False
Process_Stop ⁵	Boolean	User Writeable	No	Off/False

NOTES:

- 1 – Set this value to **True** to enable diagnostic logging from the object.
- 2 – Set this value to **True** to enable automatic startup of the Alarm Logger.
- 3 – Set the default for the **Process_FilePath** UDA to the InTouch installation directory.
e.g. **C:\Program Files (x86)\Wonderware\InTouch**
- 4 – Set the default for the **Process_Name** UDA to the InTouch Alarm Logger executable.
e.g. **wwalmlogger.exe**
- 5 – This attribute is provided to stop the process and is not part of the automated routine. Users can implement as appropriate in your applications.

3. Create (3) scripts to perform the startup of the Alarm DB Logger:

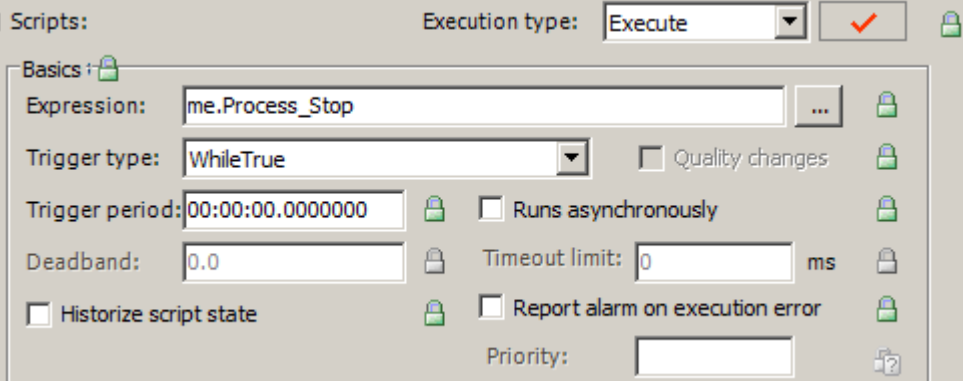
Name:	ProcessStartTrigger (Type: On Scan)
Script Info:	<input type="checkbox"/> Scripts: Execution type: OnScan
Script:	<pre>' When Process Control object goes OnScan, set flag to START the Process. If me.Process_Automate Then me.Process_Start = TRUE; if me._Debug Then LogMessage (System.String.Format("Process Control going On Scan, setting the Process Start Trigger")); Endif; else if me._Debug Then LogMessage (System.String.Format("Process Control going On Scan but Automate flag not set")); Endif; Endif;</pre>

Name:	ProcessStart (Type: Execute)
Script Info:	<div style="border: 1px solid gray; padding: 5px;"> <p>Scripts: Execution type: Execute <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/></p> <p>Basics:</p> <p>Expression: <input type="text" value="me.Process_Start"/> <input type="button" value="..."/></p> <p>Trigger type: WhileTrue <input type="checkbox"/> Quality changes</p> <p>Trigger period: <input type="text" value="00:00:00.0000000"/> <input type="checkbox"/> Runs asynchronously</p> <p>Deadband: <input type="text" value="0.0"/> <input type="checkbox"/> Timeout limit: <input type="text" value="0"/> ms</p> <p><input type="checkbox"/> Historize script state <input type="checkbox"/> Report alarm on execution error</p> <p>Priority: <input type="text"/></p> </div>
Script:	<pre> ' Process Control has been triggered to Start. Reset the Trigger flag. me.Process_Start = FALSE; if me._Debug Then LogMessage (System.String.Format("Process Control has been triggered to start process - {0}", me.Process_Name)); Endif; Dim StartFile as String; Dim ProcessNameNoExtension as String; Dim ProcessRunning as Boolean; ProcessRunning = FALSE; Dim CurrentProcess as System.Diagnostics.Process; Dim ChangeProcess as System.Diagnostics.Process; 'Get the process name from the process executable. ProcessNameNoExtension = StringLeft(me.Process_Name, StringLen(me.Process_Name)-4); if me._Debug Then LogMessage (System.String.Format("Process Name - {0}", ProcessNameNoExtension)); Endif; 'Set the location of the Process executable. StartFile = System.String.Format("{0}\{1}", me.Process_FilePath, me.Process_Name); if me._Debug Then LogMessage (System.String.Format("Location of Process Program - {0}", StartFile)); Endif; </pre>

<script continued on next page>

<script continued from previous page>

| | |
|----------------------------------|---|
| Script:
(continued...) | <pre>' Determine if the Process is already running. For Each CurrentProcess in System.Diagnostics.Process.GetProcessesByName(ProcessNameNoExtension) ProcessRunning = TRUE; if me._Debug Then LogMessage (System.String.Format("{0} Process is already running", ProcessNameNoExtension)); Endif; Next; ' If Process is not running, start it. If (NOT ProcessRunning) Then LogMessage (System.String.Format("Starting Process - {0}", ProcessNameNoExtension)); ChangeProcess = System.Diagnostics.Process.Start(StartFile); ProcessRunning = TRUE; Endif; ' update Process_Running attribute me.Process_Running = ProcessRunning;</pre> |
|----------------------------------|---|

Name:	StopProcess (Type: Execute)
Script Info:	
Script:	<pre> ' Process Control has been triggered to Stop. Reset the Trigger flag. me.Process_Stop = FALSE; if me._Debug Then LogMessage (System.String.Format("Process Control has been triggered to STOP process - {0}", me.Process_Name)); Endif; Dim ProcessNameNoExtension as String; Dim CurrentProcess as System.Diagnostics.Process; 'Get the process name from the process executable. ProcessNameNoExtension = StringLeft(me.Process_Name, StringLen(me.Process_Name)-4); if me._Debug Then LogMessage (System.String.Format("Process Name - {0}", ProcessNameNoExtension)); Endif; ' Find the running process to terminate. For Each CurrentProcess in System.Diagnostics.Process.GetProcessesByName(ProcessNameNoExtension) if me._Debug Then LogMessage (System.String.Format("{0} Process is being stopped", ProcessNameNoExtension)); Endif; CurrentProcess.Kill(); Me.Process_Running = FALSE; Next; </pre>