Tech Note 674
Monitoring an Engine and Providing Problem Notification

Topic#: 002438
OpsManage09 Session#: TS104
Created: November 2009

Introduction

This *Tech Note* contains procedures for monitoring Application Server Engines and sending emails when a problem occurs.

Application Versions

- Industrial Application Server Version 2.1 and later
- Wonderware Application Server 3.0 and later

Monitoring Engine Performance

The quickest way to check if an engine is responding is to look at Scheduler.ScanCyclesCnt and make sure it is incrementing. Each scan of the engine will increase the count.

To monitor the engine we should use an object on another engine, and preferably on another platform. The GR Platform is a good place to put this object. This example uses a User-Defined Object with four scripts. Any additional engines we want to monitor require one extra datachange script for each engine.

- Create the following UDAs
  - **AppEngine_001.Flatline** Integer
  - **AppEngine_002.Flatline** Integer
  - **MailMessage** String
  - **SendTrigger** Bool

We will reset **AppEngine_001.Flatline = 0** and **AppEngine_002.Flatline = 0** each time the **Scheduler.ScanCyclesCnt** value increments.

Script 1: ClearAppEngine_001.Flatline

Datachange script: AppEngine_001.Scheduler.ScanCyclesCnt

```
me.AppEngine_001.Flatline = 0;
```

Script 2: ClearAppEngine_002.Flatline

Datachange script: AppEngine_002.Scheduler.ScanCyclesCnt
me.AppEngine_002.Flatline = 0;

Script 3: IncrementEnginesFlatlines

This script increments the AppEngine_00x.Flatline UDAs to determine whether to send an email and/or force an Engine failover.

Periodic script every 1 second

```
me.AppEngine_001.Flatline = me AppEngine_001.Flatline + 1;
me.AppEngine_002.Flatline = me AppEngine_002.Flatline + 1;
```

The **Now()** function is used to set the current date time string into the mail message so the engineer who gets the email will know what time the error occurred in case the email was delayed before it was sent.

```
me.MailMessage = NOW () + ": ";
```

Next we check to see if the **AppEngine_001.Flatline** and or **AppEngine_002.Flatline** has incremented to **60**. If either flatline has reached 60 this tells us the Scheduler.ScanCyclesCnt has not incremented for 60 seconds and the engine or engines are in trouble.

```
If me. AppEngine_001.Flatline > 60 THEN
   me.SendTrigger = 1;
   me.MailMessage = me.MailMessage + " AppEngine_001 on WinPlatform_001 has failed to respond for 60 seconds. ";
endif;

If me. AppEngine_002.Flatline > 60 THEN
   me.SendTrigger = 1;
   me.MailMessage = me.MailMessage + " AppEngine_002 on WinPlatform_001 has failed to respond for 60 seconds. ";
endif;

me.MailMessage = me.MailMessage + "Please go to the SMC/Platform Manager to troubleshoot the problem and to restart the engine(s) that
are marked as shutdown or failed.";
```

We can limit the number of emails by looking at an attribute in our Sendmail script called ExecutionTimeStamp. In this case we will compare it to 20 minutes if it is less than 20 minutes we will reset the SendTrigger = 0 so another email is not sent until 20 minutes has expired.

```
If Now() - me.SendMail.ExecutionTimeStamp < "00:20:00:0000" then
   me.SendTrigger = 0;
endif;
```

Setting Up Emails

Our last script sets up the email parameters and sends the Email. The message can even be sent as a text message this example is for Nextel: PhoneNumber@messaging.nextel.com

Script 3: SendMail

1. Create a While True Script, with an Expression **me.SendTrigger**.
2. Check the **Runs Asynchronously** option.

3. Copy the following script into the editor:

```
logmessage (me.MailMessage);
System.Web.Mail.SmtpMail.SmtpServer = "YourMailServer.com";
System.Web.Mail.SmtpMail.Send

(
   {from: } "IAS@ABC.com",

    {to: } "JoeEngineer@ABC.com, FredEngineer@ABC.com, PhoneNumber@messaging.nextel.com",

    {subject: } "Engine Failure on PlatformName!",

    {body: } me.MailMessage
);
me.SendTrigger = 0;
```

Using the same logic in Script 3, with **IncrementEnginesFlatlines** you can add **AppEngine_001.Redundancy.ForceFailoverCmd = True**. Now the engine will failover automatically and warn you that this has occurred.

```
If me. AppEngine_001.Flatline > 60 THEN
   me.SendTrigger = 1;
   AppEngine_001.Redundancy.ForceFailoverCmd = True;
   me.MailMessage = me.MailMessage + " AppEngine_001 on WinPlatform_001 has failed to respond for 60 seconds. ForceFailoverCmd was set ";

endif;
```

R. Liddell

Tech Notes are published occasionally by Wonderware Technical Support. Publisher: Invensys Systems, Inc., 26561 Rancho Parkway South, Lake Forest, CA 92630. There is also technical information on our software products at **Wonderware Technical Support.**

For technical support questions, send an e-mail to **support@wonderware.com**.

Back to top