Tech Note 589
Common Application Server Redundancy Causes and Triggers

Topic#: 002345
Created: March 2009

Introduction

This *Tech Note* describes potential causes or triggers for a Wonderware Application Server Failover between redundant Application Engines. The SMC Log messages generated as a result of the failover are shown for each scenario.

Understanding the potential causes or triggers of a failover helps to answer the following questions. Answering these questions can result in better uptime and reduce negative impact to failovers in production environments.

1. How many times did a failover occur in my Galaxy?
2. What caused or triggered the Failover?

Application Versions

- Wonderware Application Server 3.0 SP2 and later

Key SMC Messages

Cause	**Log Message**
Engine goes into **Standby** mode.	Waiting for a redundancy status from the ObjectSyncMgr...
Engine is manually failed over or undeployed.	<***EngineName***> is shutting down objects that are deployed to it.
Engine is rebooted or powered on.	In SMC: • Logger Started. • The Bootstrap service is starting... • The NmxSvc service is running. **Note:** Cross-reference the MS Event Viewer System Log for confirmation that the machine was rebooted.

Engine cannot communicate to its redundant partner Engine	In SMC: • ResolveIPAddr couldn't resolve IP [RMC Server IP Address]. It has a valid format but cannot be resolved. • Platform [X] exceeded maximum heartbeats timeout of <Configured Timeout> ms. • The configured number of consecutirve heartbeats have been missed by engine <***X***> over the secondary message channel from partner <Platform [**X**] engine [**Y**].>
Engine does not respond to the ArchestrA Watchdog. It may be automatically restarted to recover the **aaEngine.exe** process, and started redundantly if it belongs to a redundant pair.	In SMC: • This process failed to send heartbeat and it exceeds maximum WatchdogFault... This process will be restarted as a redundant engine. • ManageRedundantProcess Process [Process ID] seems to be not responding. It has a status 12. It hasn't notified the watchdog for [Duration ex:122359] ms. The process must respond to the watchdog within [ConfigurableTimeout ex:90000] ms to be considered responsive.

How Many Times Did a Failover Occur?

The best way to count the number of failovers takes some preparation since there is no default "Failover Counter."

You must create a script for each redundant Engine. Create the scripts in an object that writes a message to the SMC Log every time the **Redundancy.Identity** attribute of its hosting engine changes between the Primary and Backup states.

Click here to download a zipped file that contains

- **$FailoverObject.aaPKG** object that contains a sample script.
- Object Viewer Watch file that lists common attributes used to monitor Redundant Engine Status.

Note: This script is a working example and will not meet your specific needs. It is included so you can customize it for your particular environment.

To use the object

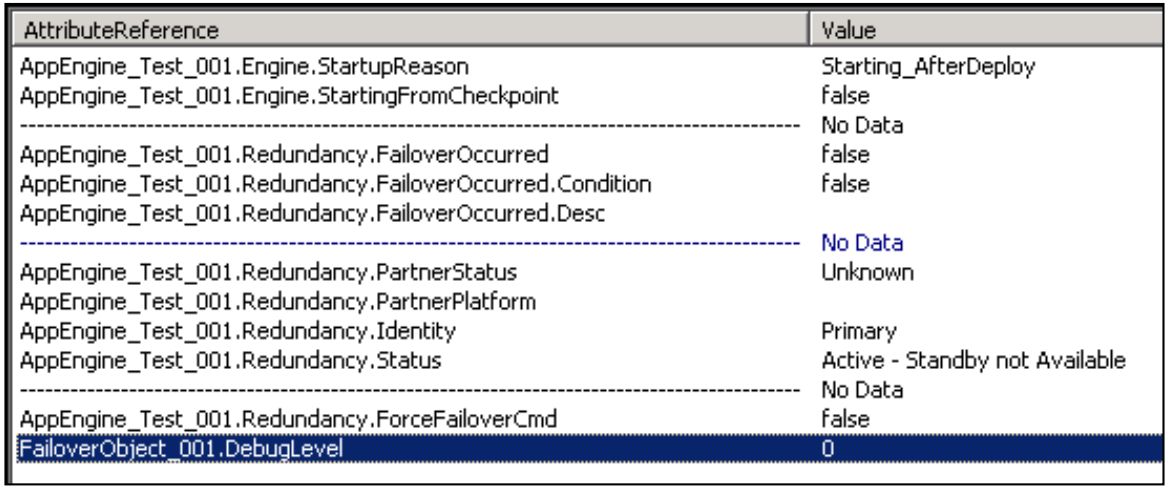
- Derive a new object instance and place it under an area hosted by a redundant Engine in your plant model. The object will automatically capture redundant-related attributes, and dump them to the SMC log during a failover or deployment.

To use the watchlist file

1. Open the watch list file with Notepad.
2. Use the Find/Replace feature to rename **AppEngine_Test_001** to the actual name of an engine in your galaxy.
3. Save and close the file.

4. Open ObjectViewer and then open the new watch window by pointing to the updated watchlist file.

   The final result should look similar to this screenshot. This assumes that you have a deployed instance of the **$Failover** object called **FailoverObject_001**.



AttributeReference	Value
AppEngine_Test_001.Engine.StartupReason	Starting_AfterDeploy
AppEngine_Test_001.Engine.StartingFromCheckpoint	false
----------	No Data
AppEngine_Test_001.Redundancy.FailoverOccurred	false
AppEngine_Test_001.Redundancy.FailoverOccurred.Condition	false
AppEngine_Test_001.Redundancy.FailoverOccurred.Desc	
----------	No Data
AppEngine_Test_001.Redundancy.PartnerStatus	Unknown
AppEngine_Test_001.Redundancy.PartnerPlatform	
AppEngine_Test_001.Redundancy.Identity	Primary
AppEngine_Test_001.Redundancy.Status	Active - Standby not Available
----------	No Data
AppEngine_Test_001.Redundancy.ForceFailoverCmd	false
FailoverObject_001.DebugLevel	0

Figure 1: Watchlist in Object Viewer

What Causes or Triggers a Failover?

A failover occurs when the Standby Engine in a redundant configuration can no longer communicate with the Active Engine of its redundant partner. It is important to note the significance of the wording in the above sentence. The **hearbeat** mechanism that is constantly monitored across the RMC Chanel is between **Application Server Engines** and not just the physical availability of the Server hardware.

For example, a redundant Platform server can be up and running, and even the network is running. The **Active** Redundant Engine is overloaded with Scan Overruns and does not respond in time to hearbeat requests from its redundant partner.

This condition triggers a failover condition and the Standby engine goes to **Active**.

Other variables could also effect this scenario, such as network communication issues or an unresponsive server due to faulty or starving resources (hard drive space, memory and CPU).

Failover Scenario Examples

The following scenarios include these elements:

- All scenarios start with the Active Partner.
- The Active Partner is **GR Platform** and the Standby Partner is **Remote Platform**.
- The GR Platform Server Name is: WWNESTEST (This is a Windows 2003 R2 SP2 OS)

- The Remote Platform Server Name is: NACKT_WINXP_E (This is a Windows XP SP2 OS)

Note: The Operating Systems used in this *Tech Note* are different from one another (Win2003 vs WinXP). This is for illustration purposes only. Best Practice dictates to always use the same OS between redundancy-enabled engines.

Logging the Manual Force Failover Set

On the GR Node, a manual force failover is initiated by setting the engine attribute **[Engine_Name].Redundancy.ForceFailoverCmd** to TRUE.

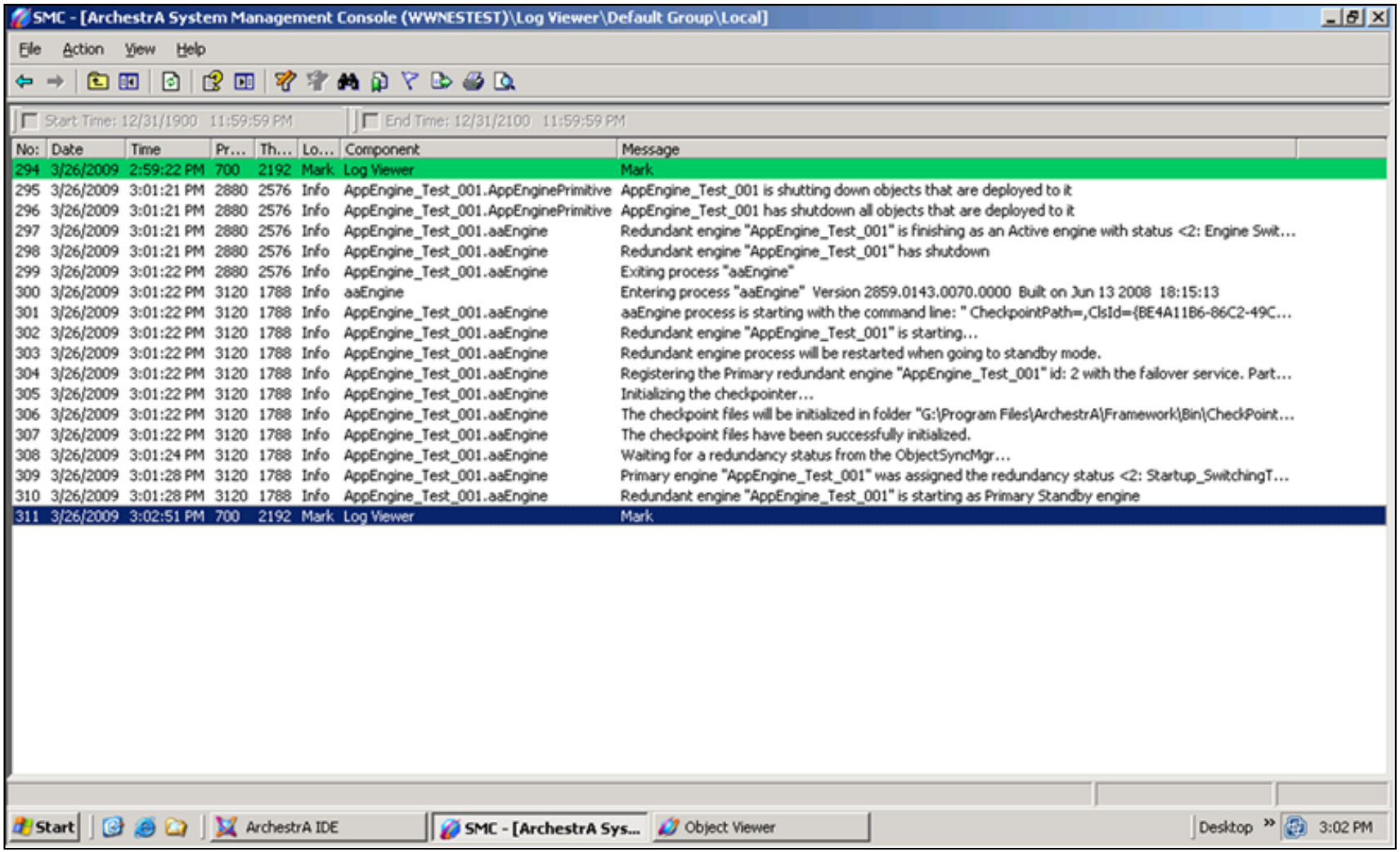


Figure 2: Manual Force Failover: [ENGINE_NAME].REDUNDANCY.FORCEFAILOVERCMD IS SET TO TRUE

The following graphic shows the Remote Backup Node as it becomes active.

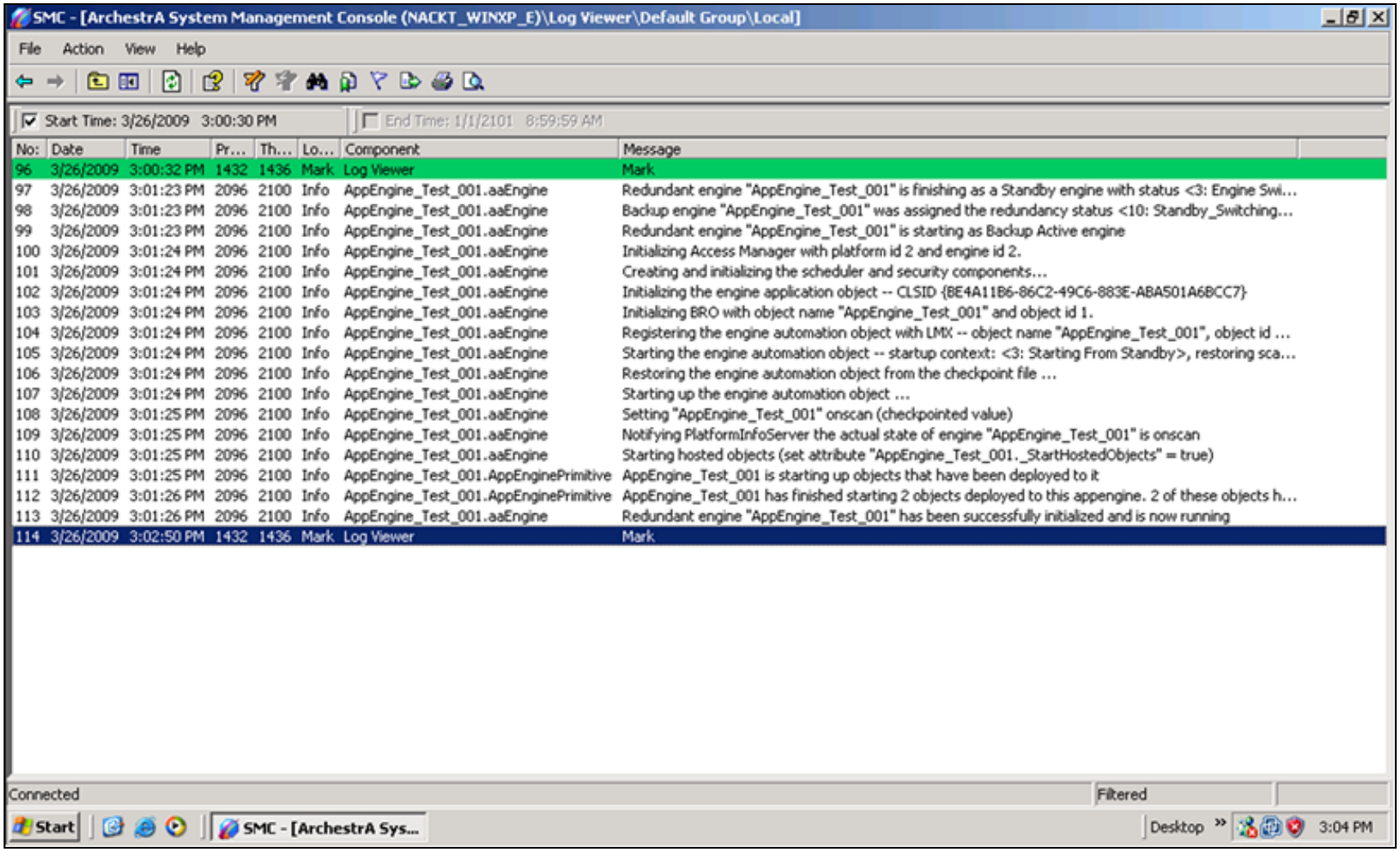


Figure 3: REDUNDANT ENGINE ON REMOTE PLATFORM IS [...] "SUCCESSFULLY INITIALIZED AND IS NOW RUNNING."

Logging the Active Partner Crash/Power Cut/Communication Loss Failover

In this scenario, the following events occur:

- The GR Platform suddenly becomes unavailable either due to a power failure, network communication loss, or some kind of system crash.
- The Redundant Engine fails over to the Remote Platform.
- The GR Platform is restored and becomes available.

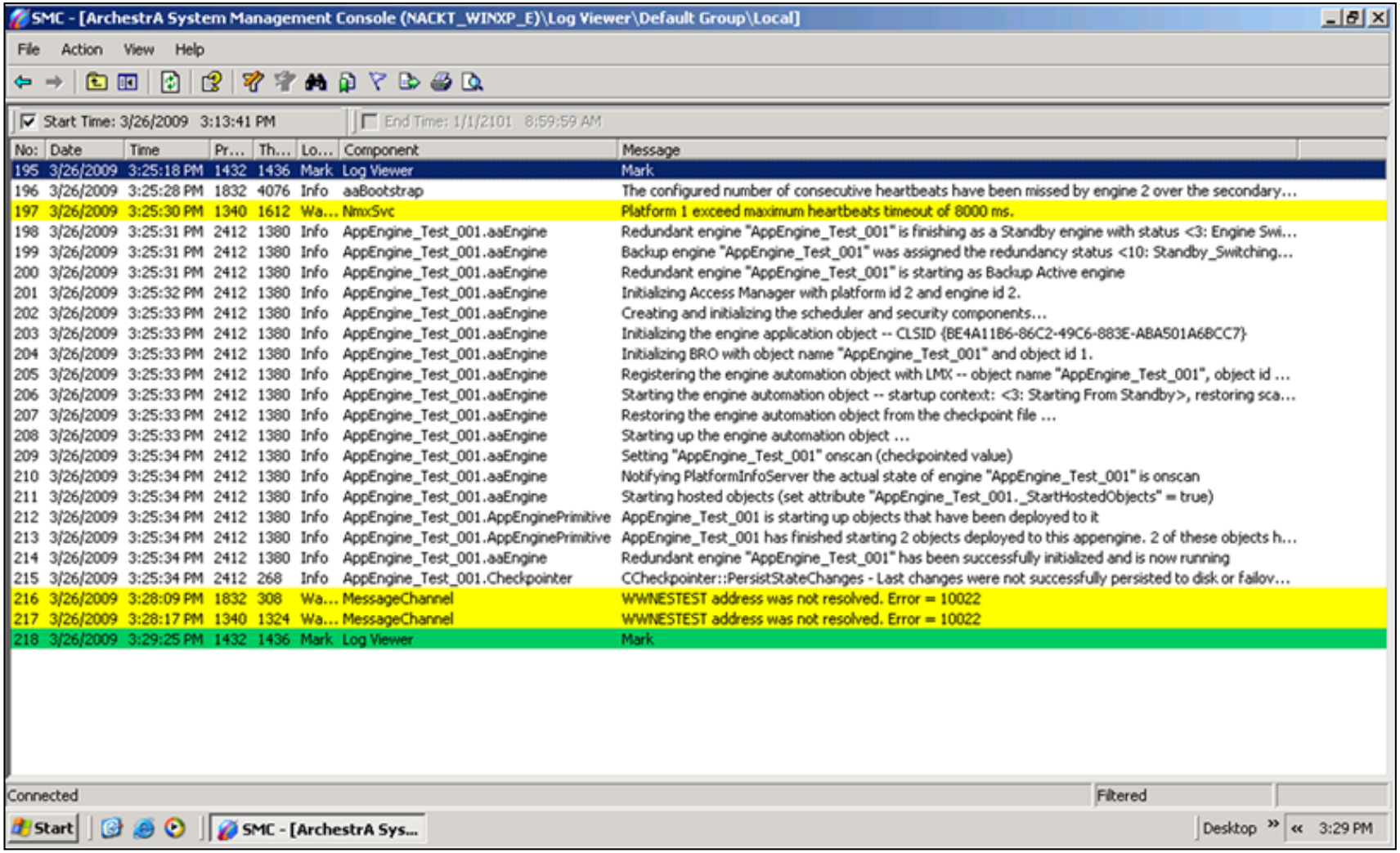


Figure 4: Platform Detects That It can No Longer Communicate With Its Redundant Partner

Log messages show the restart from checkpoint:

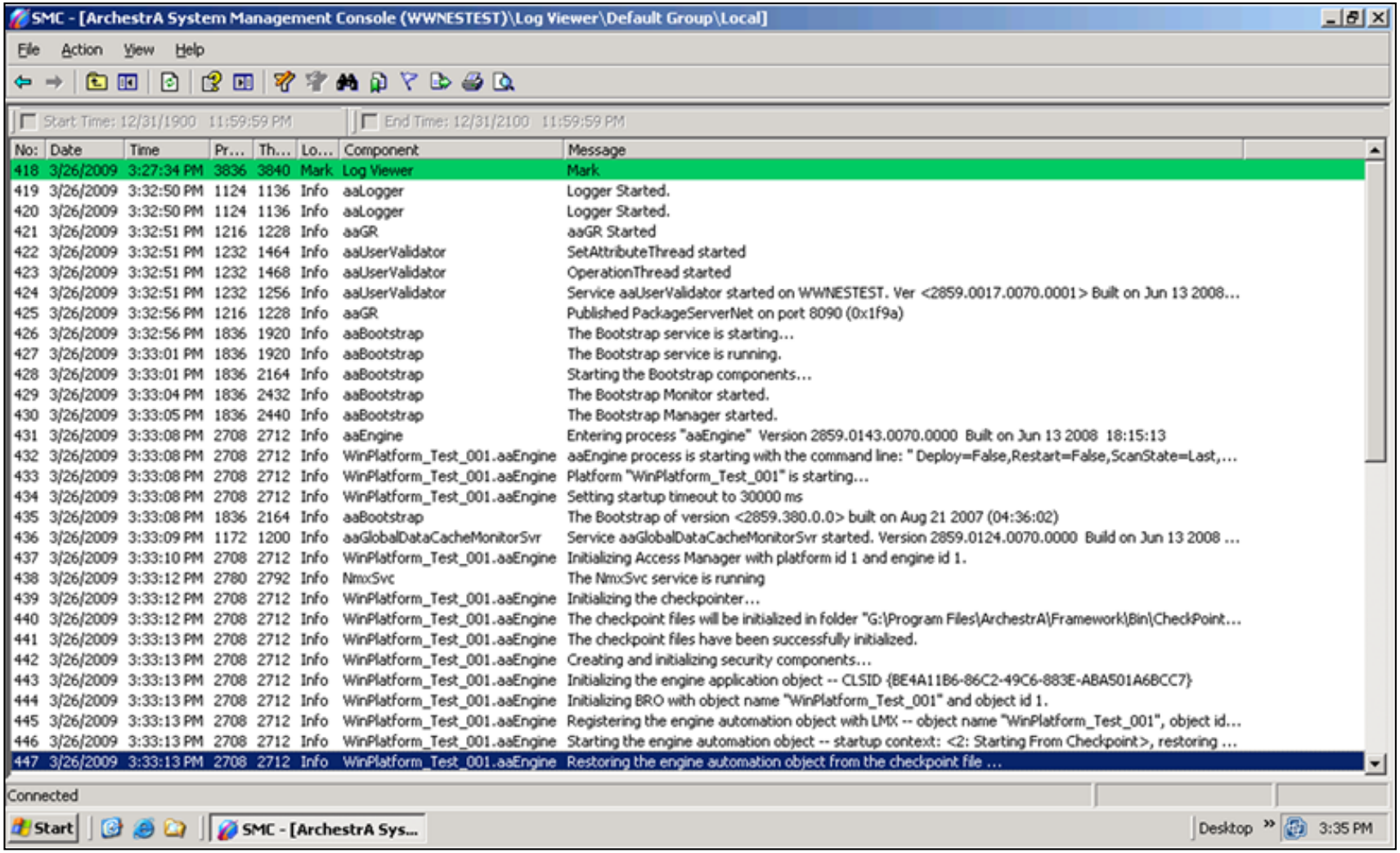


Figure 5: GR Platform Recovers and Becomes Available

Figure 6 (below) shows the additional logged messages in the SMC from the GR Platform after becoming available again:

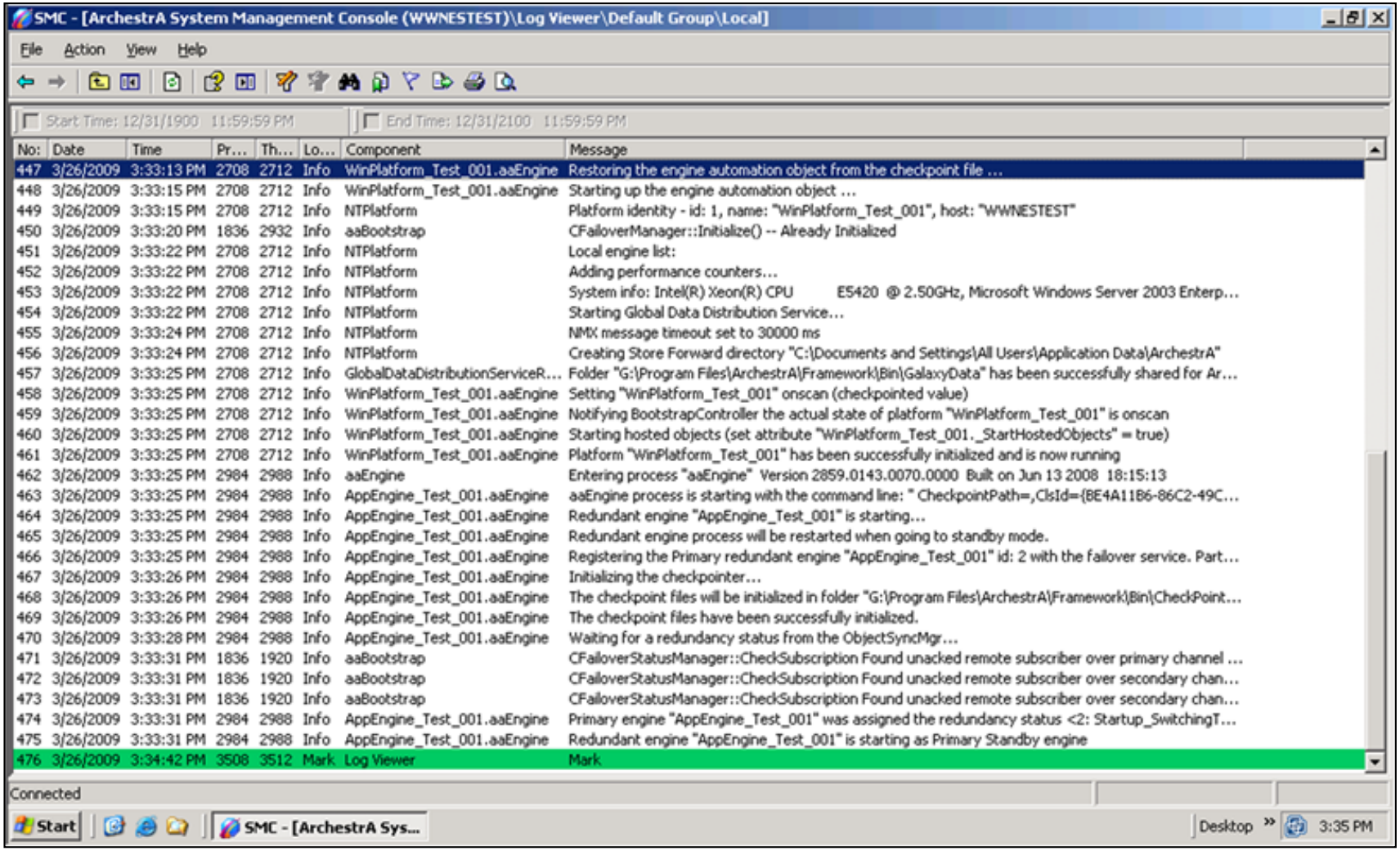


Figure 6: Continuation of Log File from Figure 5 (Above)

Figure 7 (below) shows the GR Platform's MS Event Viewer System Application Log. Here we are highlighting the event that captured the system shutdown.

This event complements the Remote Platform's SMC Log messages indicating a communication loss with its redundant partner.

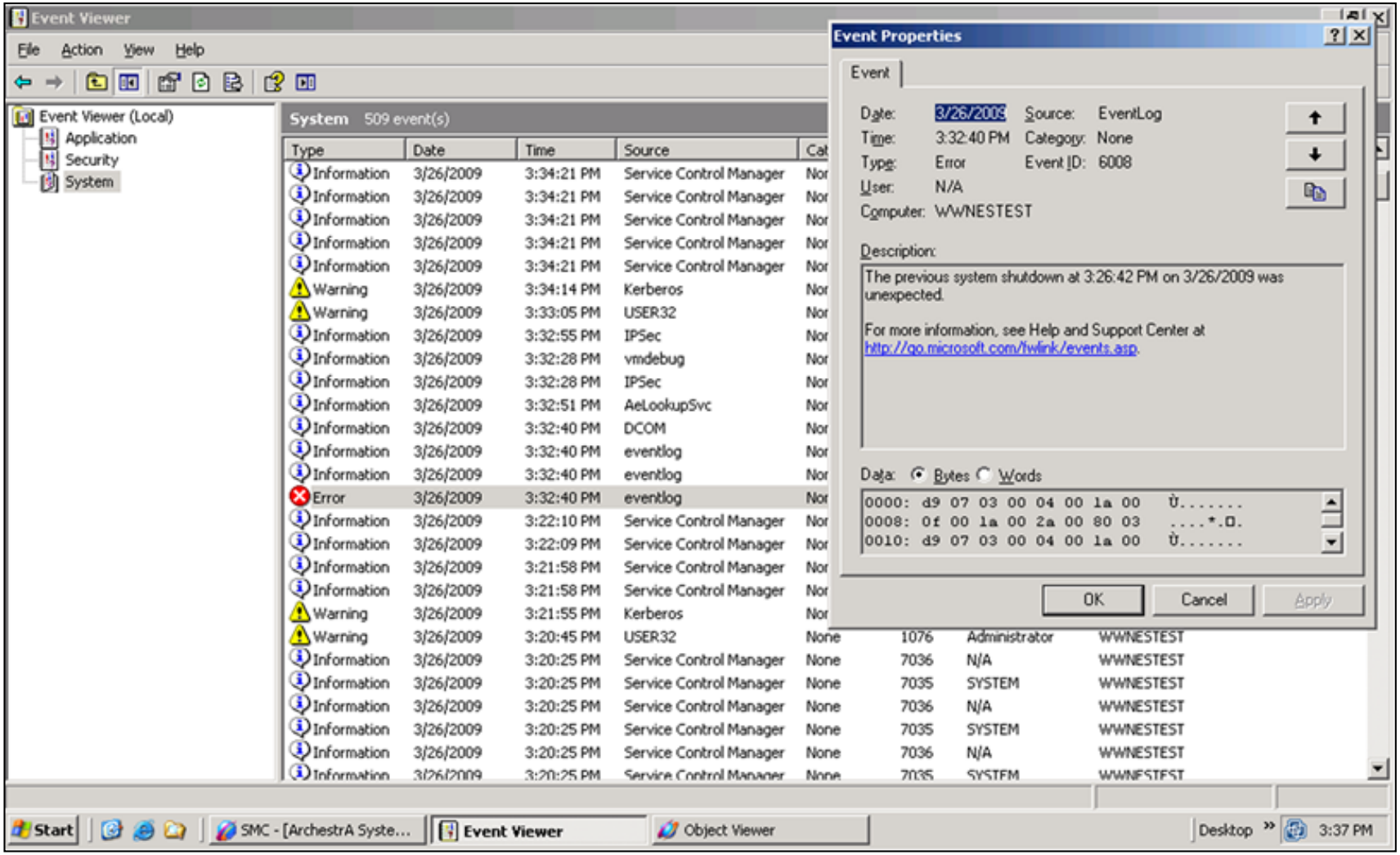


Figure 7: MS Event Log/system at Failover

Undeployed Hosting Engine

In this scenario, the GR Platform Hosts the Active Engine. It is undeployed and the failover is triggered.

Messages logged on the GR PLATFORM:

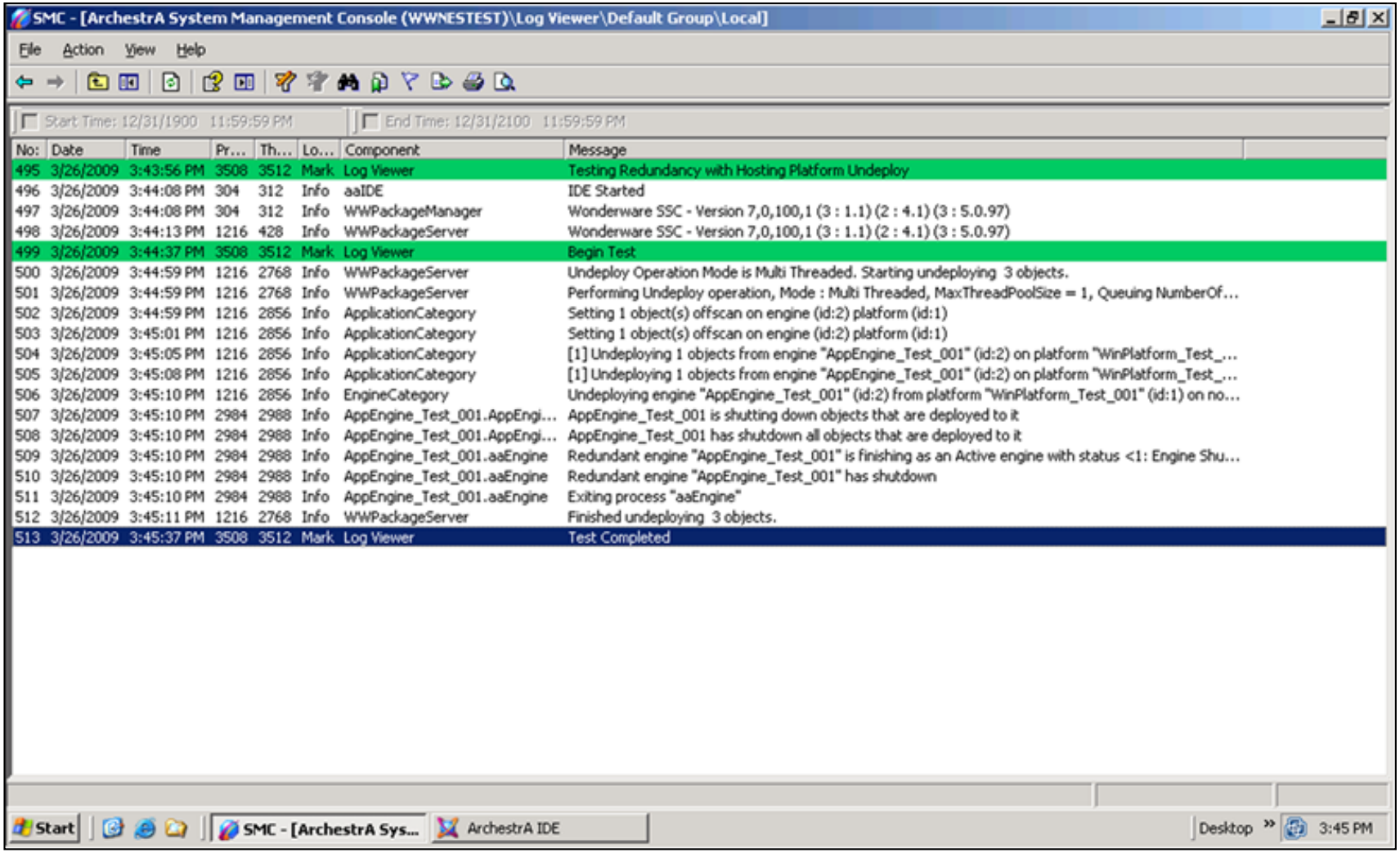


Figure 8: GR Platform Log Messages

On the remote Platform, its platform was undeployed. This was the hosting primary engine of the redundant pair.

The messages look like the following graphic.

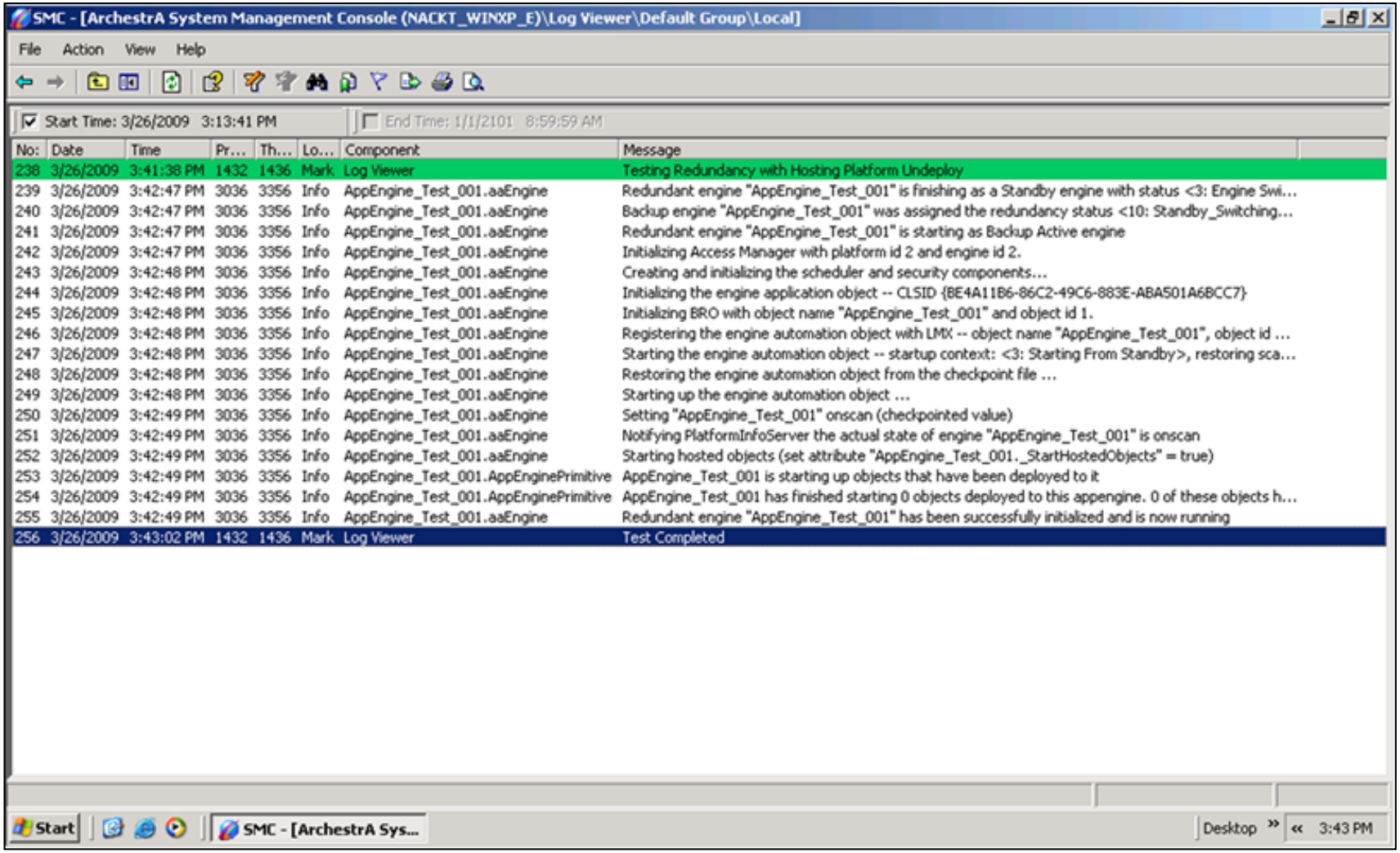


Figure 9: Remote Platform / Hosting Engine Log View from the SMC

Prebuilt InTouch® ArchestrA Graphics for Redundancy

A predefined ArchestrA Graphic for Engine Redundancy Statistics is provided with the standard install of Application Server 3.0.

The graphic is called **AppEngineRedundancyDisplay** and is available from the Graphic Toolbox in the IDE.

In the IDE, from the Graphic Toolbox, navigate to **ArchestrA Symbol Library > Displays**.

Figure 9 (below) shows the graphic when implemented in Runtime.

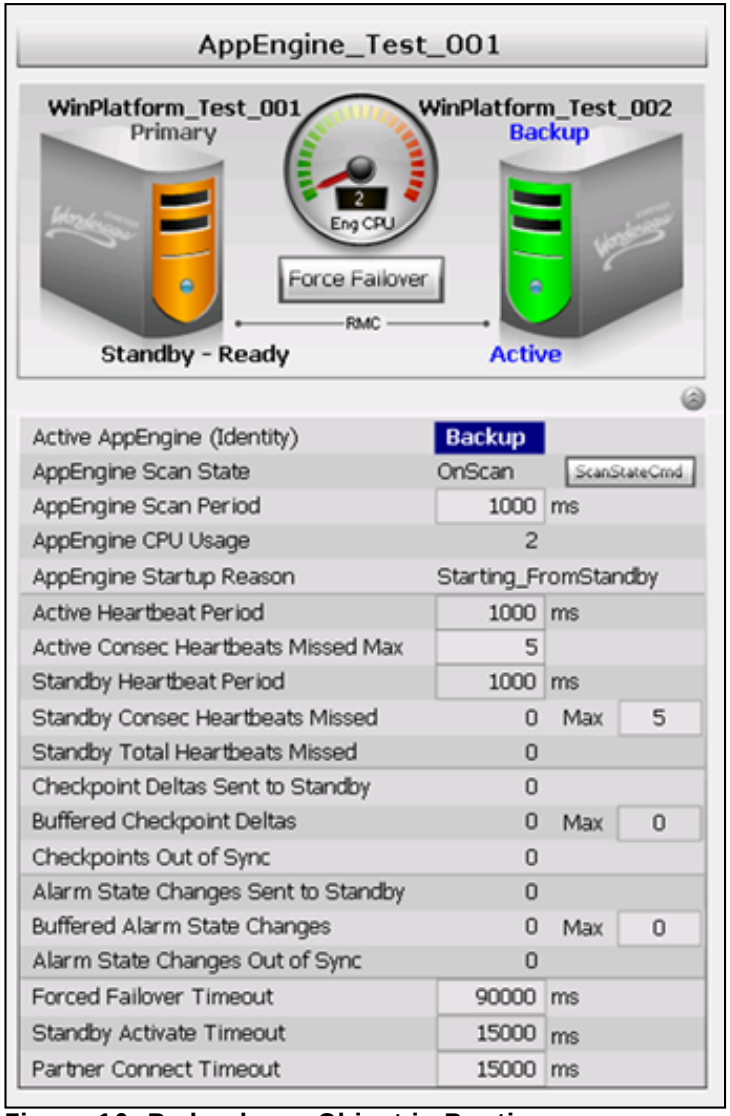


Figure 10: Redundancy Object in Runtime

To place the graphic in a template or derived instance of an engine:

1. From the **Graphics** tab of AppEngine editor, add a new Local Graphic.
2. Open the graphic and click the **EMBED GRAPHIC** button.
3. Browse to this symbol under the Graphic Toolbox tree view and select it.
4. Once it is on your graphic editor as an embedded symbol, you are done.
5. Save the symbol and close. *Do not* convert the embedded symbol to a group.

Note: The Graphic Toolbox contains several other ArchestrA symbols that show useful information about Engines and Platforms.

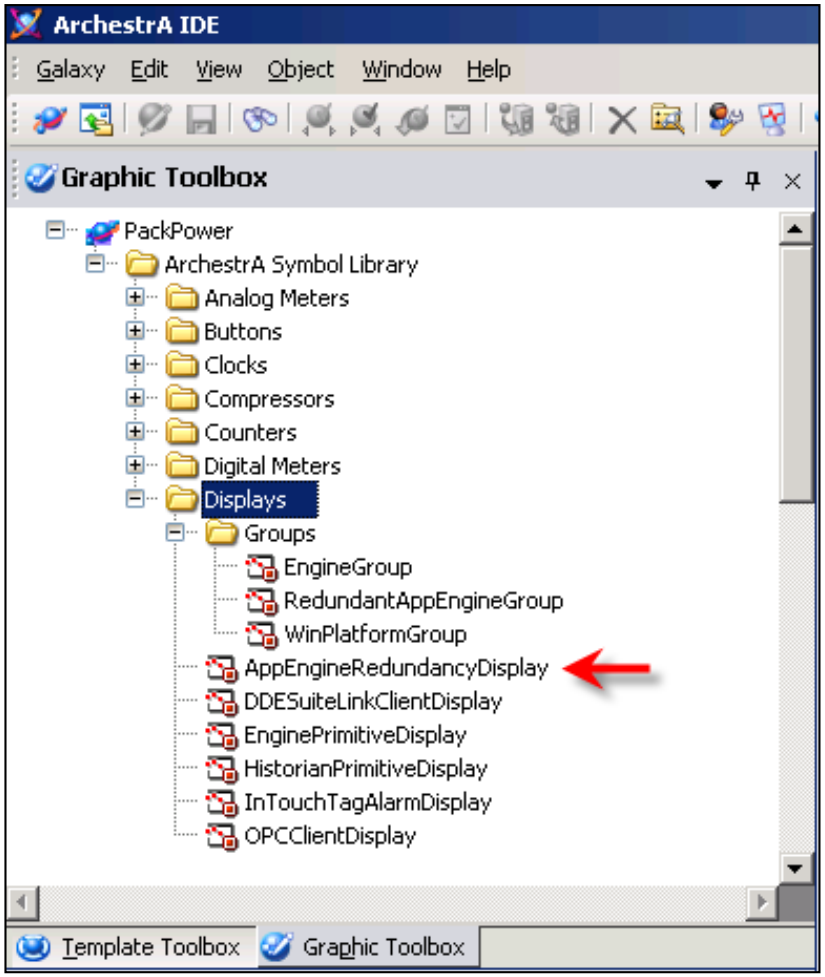


Figure 11: Diagnostic Symbols in Graphic Toolbox

References

For more information on implementing & optimizing redundancy-enabled engines, please refer to the following TechNotes:

- **Tech Note 368 Network Setup for AppEngine Redundancy**
- **Tech Note 401 Fine-Tuning AppEngine Redundancy Settings**
- **Tech Note 478 Industrial Application Server Platform Deployment Checklist**

- **Tech Note 554 Rebuilding/Replacing a Redundant AppObject Server Node**

S. Kermani

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