Tech Note 816
Tips for Troubleshooting AppEngine Redundancy-Related Issues

Topic#: 002606
Created: January 2012

Introduction

AppEngine Redundancy is designed to provide high availability of an Engine and its hosted application objects in Runtime, against either a single engine failure or any type of system failure. When failure occurs, high availability is only possible if the runtime redundancy pair (Active and Standby) are in a prepared/ready state, and the failover process completes successfully.

This *Tech Note* provides some tips for troubleshooting AppEngine failover-related issues.

Application Versions

- Application Server 3.1 and later.

AppEngine Automatically Fails Over on Reboot of One of the Redundant Pair

Issue

One of the redundant pair Platform/computers is shut down for some reason. All redundant AppEngines fail over and run as **Active** on its partner platform/computer.

After rebooting the shutdown Platform/computer, all redundant AppEngines on that platform should start up and run as **Standby** if there is no failover command from either scripts or from an operator's manual entry.

However, sometimes a redundant AppEngine on the newly-rebooted Platform/computer starts up, goes to run as **Active**, and relegates its partner AppEngine (which is already running as **Active**) to **Standby**, even when there is no failover command.

Resolution

The **Maximum time to discover partner** AppEngine Redundancy configuration setting (Figure 1 below) or the **PartnerConnectTimeout** setting for the AppEngine's Redundancy Primitive (Figure 2 below) are configuration settings used to determine how long to wait before the AppEngine attempts to connect to the failover partner before setting the failover partner state to **Unknown**. The default setting is **15,000** ms. When this setting is too short, the Automatic Failover on Reboot issue is likely to occur.

To resolve this issue, increase this setting. The following explains the relationship between this setting and the issue.

Cause

After the Bootstrap (service) initiates a start of an AppEngine with redundancy enabled, before deciding its own redundancy role (**Active** or **Standby**) to assume, the AppEngine's Redundancy Primitive immediately starts to contact its partner Platform and its partner AppEngine through both primary and RMC network channels to get its partner' current status. At the same time, a timer is started to count down with an interval value set according to "Maximum time to discover partner" (aka "PartnerConnectTimeout" in runtime attribute).

Within this time period

- If the Redundancy Primitive of the AppEngine receives a response (through either channel) from its partner Platform and AppEngine, it calculates to derive its partner AppEngine status, and begins to adjust its own role as either **Active** or **Standby** based on the derived partner's current status.
- If the Redundancy Primitive fails to receive any response from its partner platform or AppEngine within this time period, the AppEngine assigns its partner AppEngine status as **Unknown** and elevates its own role to **Active**.

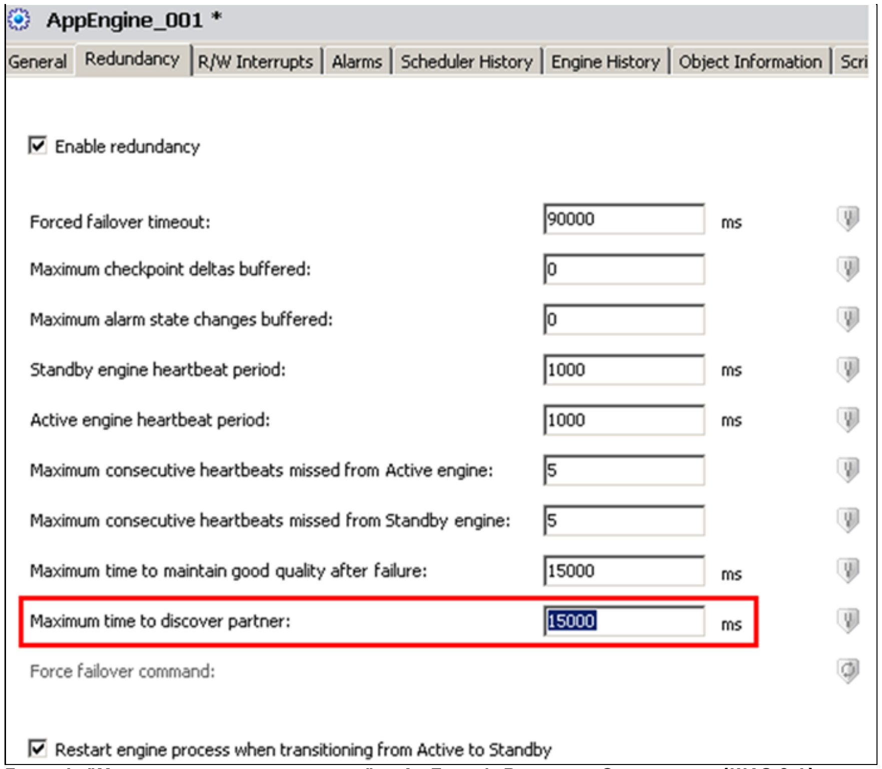
The length of time to take from start of an AppEngine to receiving a response from its partner AppEngine can vary, and depends on many different factors. One of the most important factors is the load of the AppEngine itself and load of its hosting platform (too many other AppEngines on the same platform). The higher the load, the longer it takes to establish communication between the AppEngine and its partner platform.

Therefore, it is completely possible that even when its partner platform and AppEngine are up and running, the AppEngine's Redundancy Primitive fails to receive a response from its partner AppEngine within the time period set by the timer. When this happens, the redundancy pair status ends up in the condition **Active-Active**. After an extended time period, when the pair eventually reaches each other, one of pair has to relegate itself from Active role to Standby. By design, in this situation the Primary AppEngine will remain as Active, the Backup AppEngine must back down and switch to **Standby**.

Because of many runtime factors, your appropriate setting for this redundancy parameter should be decided by trial-and-error in a testing environment. *

***Note:** If **Maximum time to discover partner** (**PartnerConnectTimeout** in the runtime attribute) is modified, ensure that **WatchdogStartupTimeout** and **WatchdogShutdownTimeout** are adjusted accordingly.

WatchdogStartupTimeout and **WatchdogShutdownTimeout** should be greater than the **PartnerConnectTimeout** setting. In this context it is recommended that **WatchdogStartupTimeout** and **WatchdogShutdownTimeout** values should be more than double the **PartnerConnectTimeout** value.



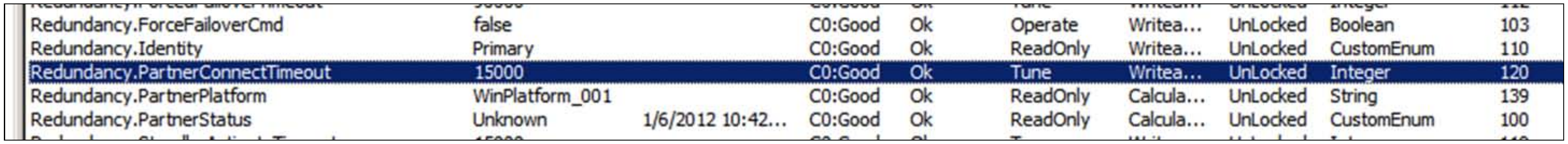
AppEngine_001 *

☑ Enable redundancy

Forced failover timeout:	90000	ms
Maximum checkpoint deltas buffered:	0	
Maximum alarm state changes buffered:	0	
Standby engine heartbeat period:	1000	ms
Active engine heartbeat period:	1000	ms
Maximum consecutive heartbeats missed from Active engine:	5	
Maximum consecutive heartbeats missed from Standby engine:	5	
Maximum time to maintain good quality after failure:	15000	ms
Maximum time to discover partner:	15000	ms
Force failover command:		

☑ Restart engine process when transitioning from Active to Standby

Figure 1: "Maximum time to discover partner" in AppEngine's Redundancy Configuration (WAS 3.1)



Redundancy.ForceFailoverCmd	false		C0:Good	Ok	Operate	Writea...	UnLocked	Boolean	103
Redundancy.Identity	Primary		C0:Good	Ok	ReadOnly	Writea...	UnLocked	CustomEnum	110
Redundancy.PartnerConnectTimeout	15000		C0:Good	Ok	Tune	Writea...	UnLocked	Integer	120
Redundancy.PartnerPlatform	WinPlatform_001		C0:Good	Ok	ReadOnly	Calcula...	UnLocked	String	139
Redundancy.PartnerStatus	Unknown	1/6/2012 10:42...	C0:Good	Ok	ReadOnly	Calcula...	UnLocked	CustomEnum	100

Figure 2: "PartnerConnectTimeout" Attribute in Redundant AppEngine Redundancy Primitive

AppEngine Ends in "Startup Failed" or "Shutdown Failed" State After Failover Initiated

Issue

After a failover command is initiated (either from an automatic failover condition or manually triggered), one or both of the AppEngine pair can end up in the **Startup Failed** or **Shutdown Failed** state.

Resolution

When an AppEngine is called to start or shut down by the Bootstrap (service), the startup or shutdown process is constantly monitored by the Bootstrap attribute called **Watchdog Timer**. If the startup or shutdown is not completed within a time period, the AppEngine is prematurely killed. Restart or Shutdown is then called.

If startup and shutdown fails again, the AppEngine is marked as **Startup Failed** and **Shutdown Failed**, and stays suspended in that state. By default, the startup and shutdown timeout period is **30** seconds for Application Server 3.1 or prior. In ASP 2012 (WAS 3.5), the default setting is increased to 5 minutes for both operations.

The length of time it takes to complete an AppEngine startup and shutdown varies, and depends on many factors. The most important factors include the load of the AppEngine and overall CPU usage condition on that platform. It is often found that the default startup and shutdown timeout setting is not adequate as the load of platform or the AppEngine is growing.

The timeout settings for startup and shutdown can be adjusted by adding the following registry keys:

For 32-bit Operating Systems:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\ArchestrA\Framework\Platform]
"WatchdogStartupTimeout"=dword:000493e0
"WatchdogShutdownTimeout"=dword:000493e0
```

For 64-bit Operating Systems:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\ArchestrA\Framework\Platform]
"WatchdogStartupTimeout"=dword:000493e0
"WatchdogShutdownTimeout"=dword:000493e0
```

Notes

- The maximum value for both settings is **300,000** ms (5 minutes).
- The value for the **WatchdogStartupTimeout** and **WatchdogShutdownTimeout** settings should be at least greater than the value of **Maximum time to discover partner** (**PartnerConnectTimeout**) in the AppEngine Redundancy configuration settings. It is recommended that these settings should be more than double the value of the **PartnerConnectTimeout** for a better performance.

AppEngine's PartnerStatus Stays in "Standby – Not Ready"

Issue

After a failover is initiated, the Standby AppEngine will elevate its status to **Active**. Meanwhile, the Active AppEngine will relegate its current Active status to **Standby**. For the new Active AppEngine, the Redundancy Primitive's **PartnerStatus** attribute should go through various substates of the **Standby** status, eventually reaching **Standby – Ready** as final state of the failover process, in order to successfully complete the failover process.

However, **PartnerStatus** sometimes stays in **Standby – Not Ready** at the end of a failover process, and stops moving forward.

Resolution

PartnerStatus is a calculated attribute of a Redundancy Primitive. Its value is derived from a calculation based on the conditions of its partner Platform and AppEngine obtained at the time. This means that it changes as the failover process moves along.

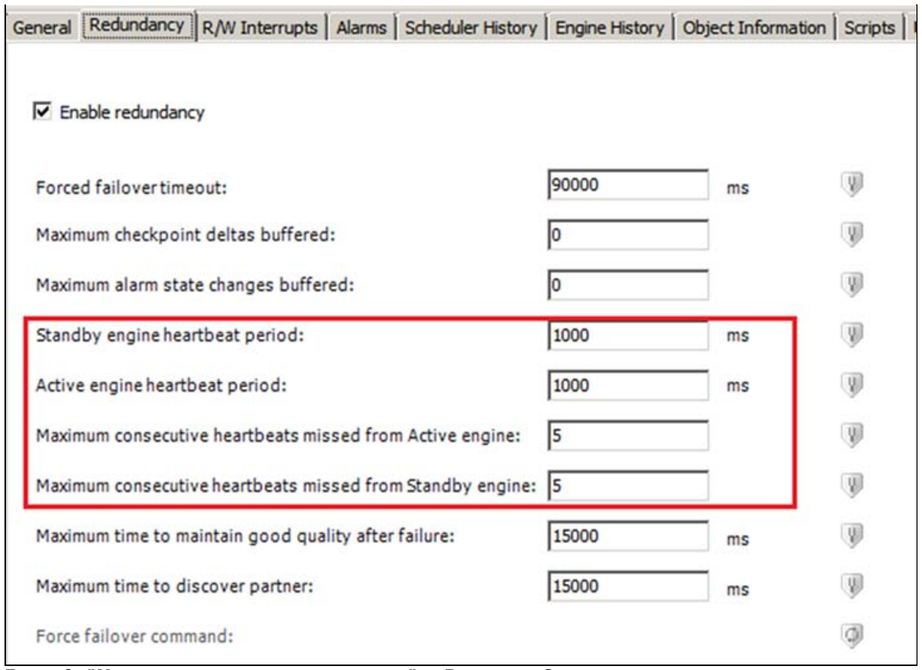
The **Standby – Not Ready** status can be derived from one of the following conditions during the transition:

1. Heartbeats missed via RMC.
2. Heartbeats missed via Primary ArchestrA Network.
3. Either one of the redundant platforms cannot be reached from Active AppEngine (role after failover).
4. Either one of the redundant platforms cannot be reached from Standby AppEngine (role after failover).
5. Standby AppEngine (role after failover) is in process of synchronizing with the Active AppEngine (role after failover) for Object Information, Checkpointing Data, and Subscription Information.

Cause

The **Heartbeats Missed** condition is usually generated by either system performance or network-related issues. The root cause to **Heartbeats Missed** condition should be addressed to alleviate **Standby – Not Ready** status.

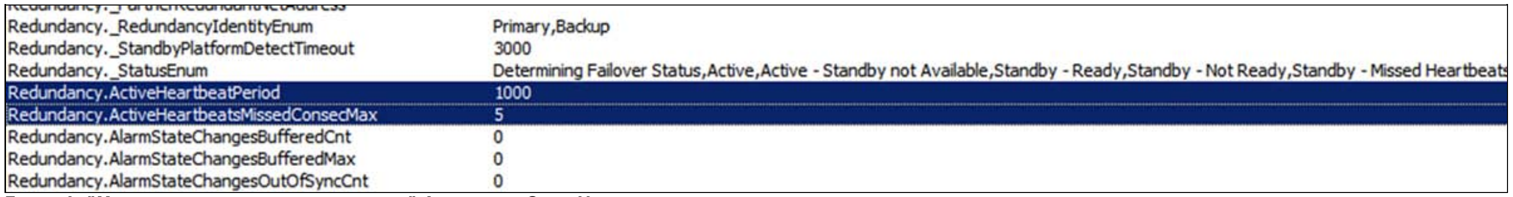
Adjusting **Maximum consecutive heartbeats missed** (Figure 3 below) should be only considered when you are interested in temporarily relieving the **Standby – Not Ready** status condition. Note that this adjustment should be done through the attribute in ObjectViewer so no redeploy is required. (Figure 4 below).



☑ Enable redundancy

Forced failover timeout:	90000	ms
Maximum checkpoint deltas buffered:	0	
Maximum alarm state changes buffered:	0	
Standby engine heartbeat period:	1000	ms
Active engine heartbeat period:	1000	ms
Maximum consecutive heartbeats missed from Active engine:	5	
Maximum consecutive heartbeats missed from Standby engine:	5	
Maximum time to maintain good quality after failure:	15000	ms
Maximum time to discover partner:	15000	ms
Force failover command:		

Figure 3: "Maximum consecutive heartbeats missed" in Redundancy Configuration



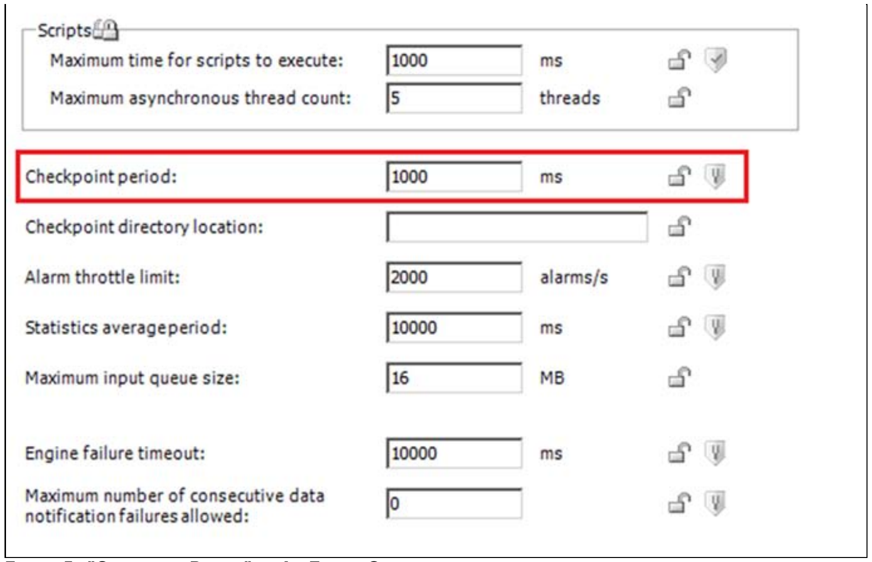
Redundancy._RedundancyIdentityEnum	Primary,Backup
Redundancy._StandbyPlatformDetectTimeout	3000
Redundancy._StatusEnum	Determining Failover Status,Active,Active - Standby not Available,Standby - Ready,Standby - Not Ready,Standby - Missed Heartbeats
Redundancy.ActiveHeartbeatPeriod	1000
Redundancy.ActiveHeartbeatsMissedConsecMax	5
Redundancy.AlarmStateChangesBufferedCnt	0
Redundancy.AlarmStateChangesBufferedMax	0
Redundancy.AlarmStateChangesOutOfSyncCnt	0

Figure 4: "Maximum consecutive heartbeats missed" Attribute in ObjectViewer

The Standby AppEngine must completely synchronize Object Information, Checkpointing Data, and Subscription Information from the Active AppEngine before it is considered **Ready**. It is possible for the Standby AppEngine to be overwhelmed when catching up with information updates, particularly when there are a large number of checkpointing attributes, and system performance is a stringent condition.

To alleviate this condition, you should consider one or more of the following measures:

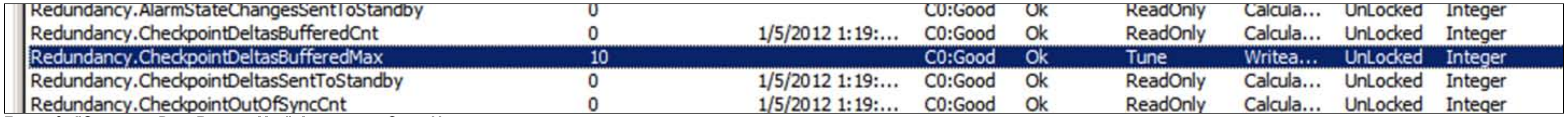
- Increase the AppEngine **Checkpoint period** value - a Redeploy needed (Figure 5 below).



Scripts		
Maximum time for scripts to execute:	1000	ms
Maximum asynchronous thread count:	5	threads
Checkpoint period:	1000	ms
Checkpoint directory location:		
Alarm throttle limit:	2000	alarms/s
Statistics averageperiod:	10000	ms
Maximum input queue size:	16	MB
Engine failure timeout:	10000	ms
Maximum number of consecutive data notification failures allowed:	0	

FIGURE 5: "CHECKPOINT PERIOD" IN APPENGINE CONFIGURATION

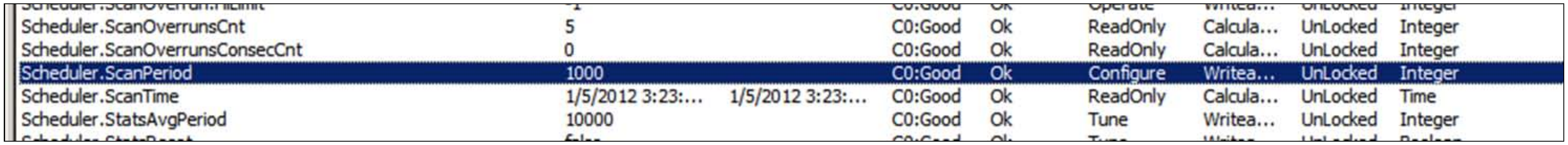
- Reduce the **CheckpointDeltaBufferedMax** value if it is non-zero (Figure 6 below).



Redundancy.AlarmStateChangesSentToStandby	0		C0:Good	Ok	ReadOnly	Calcula...	UnLocked	Integer
Redundancy.CheckpointDeltasBufferedCnt	0	1/5/2012 1:19:...	C0:Good	Ok	ReadOnly	Calcula...	UnLocked	Integer
Redundancy.CheckpointDeltasBufferedMax	10		C0:Good	Ok	Tune	Writea...	UnLocked	Integer
Redundancy.CheckpointDeltasSentToStandby	0	1/5/2012 1:19:...	C0:Good	Ok	ReadOnly	Calcula...	UnLocked	Integer
Redundancy.CheckpointOutOfSyncCnt	0	1/5/2012 1:19:...	C0:Good	Ok	ReadOnly	Calcula...	UnLocked	Integer

FIGURE 6: "CHECHPOINTDELTABUFFEREDMAX" ATTRIBUTE IN OBJECTVIEWER

- Before a failover starts, increase the AppEngine's **Scheduler.ScanPeriod** value using ObjectViewer (Figure 7 below). This can also help the Standby AppEngine catch up with Active AppEngine in synchronization to achieve **Standby – Ready** state for its PartnerStatus.



Scheduler.ScanOverrunsCnt	5		C0:Good	Ok	ReadOnly	Calcula...	UnLocked	Integer
Scheduler.ScanOverrunsConsecCnt	0		C0:Good	Ok	ReadOnly	Calcula...	UnLocked	Integer
Scheduler.ScanPeriod	1000		C0:Good	Ok	Configure	Writea...	UnLocked	Integer
Scheduler.ScanTime	1/5/2012 3:23:...	1/5/2012 3:23:...	C0:Good	Ok	ReadOnly	Calcula...	UnLocked	Time
Scheduler.StatsAvgPeriod	10000		C0:Good	Ok	Tune	Writea...	UnLocked	Integer

FIGURE 7: APPENGINE'S "SCANPERIOD" ATTRIBUTE IN OBJECTVIEWER

AppEngine's PartnerStatus is "Unknown"

Issue

Active AppEngine's redundancy attribute **PartnerStatus** remains **Unknown**.

Resolution

This redundancy status is invoked when its partner platform cannot be reached. This could be the result of platform shutdown/not deployed, or completely interrupted Primary and RMC networks.

AppEngine's PartnerStatus is "Active – Not Available"

Issue

Active AppEngine's redundancy attribute **PartnerStatus** remains **Active – Not Available**.

Resolution

If both platforms are deployed and running On Scan, verify the following:

- The network IP resolution to both the platforms' Network Address
- Check the ArchestrA Network configuration to ensure its NIC is in first position in the Network Binding Order.
- Check if redundancy message channel is correctly configured on the platform.

Cause

This redundancy status is invoked when the Standby AppEngine (vis-à-vis to the current Active AppEngine)

- Can only communicate (but is missing heartbeats) via the RMC network,
- Fails to reach the Active AppEngine via the Primary network, and
- No other Platform can communicate with the Platform that hosts the Active AppEngine.

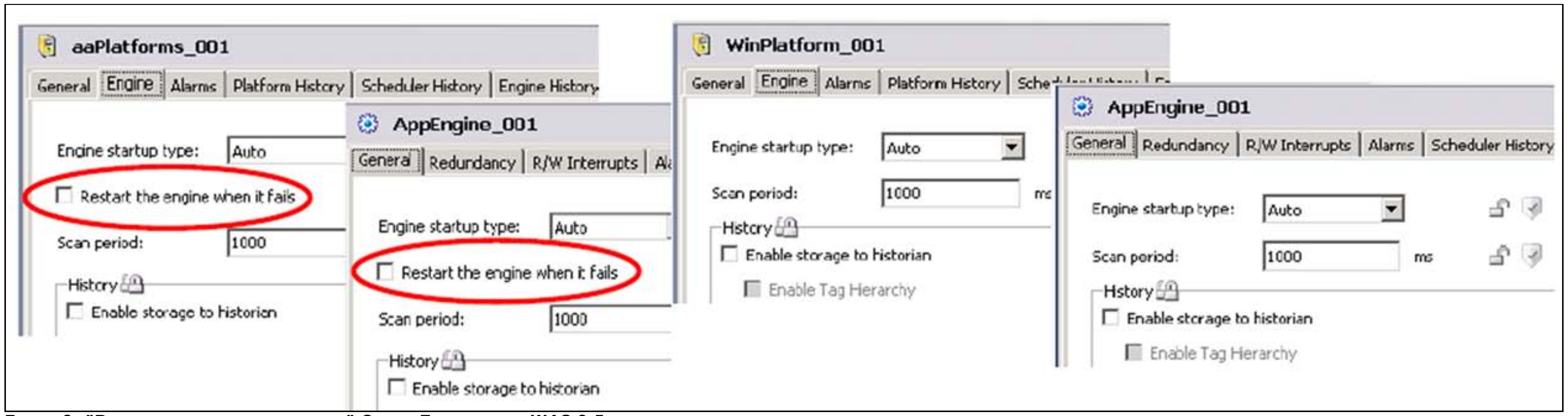
Redundancy Improvements in ASP 2012 (WAS 3.5) over Application Server 3.1 or Prior

The following design changes are implemented in WAS 3.5. The changes should improve the overall performance of AppEngine Redundancy.

Eliminated "Restart the Engine When it Fails" Option

This option is eliminated in both the **Platform (Engine tab)** and the **AppEngine (General tab)** Configuration.

AppEngine is now always restarted when crash/hang condition is detected. In WAS 3.1 or before, if the **Restart the engine when it fails** option is left unchecked, when an AppEngine crashes, its redundant partner will detect the condition and fail over. But it can never fail back because of an orphaned partner due to the AppEngine crash.



aaPlatforms_001 — General | Engine | Alarms | Platform History | Scheduler History | Engine History

Engine startup type: Auto
☐ Restart the engine when it fails
Scan period: 1000
History
☐ Enable storage to historian

AppEngine_001 — General | Redundancy | R/W Interrupts | Al

Engine startup type: Auto
☐ Restart the engine when it fails
Scan period: 1000
History
☐ Enable storage to historian

WinPlatform_001 — General | Engine | Alarms | Platform History | Sche

Engine startup type: Auto
Scan period: 1000 ms
History
☐ Enable storage to historian
☐ Enable Tag Hierarchy

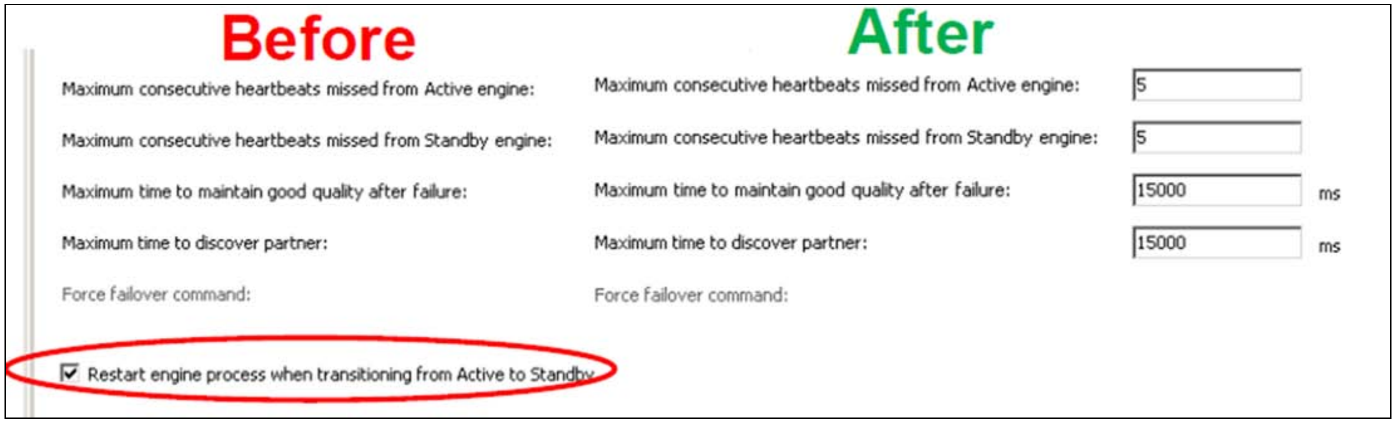
AppEngine_001 — General | Redundancy | R/W Interrupts | Alarms | Scheduler History

Engine startup type: Auto
Scan period: 1000 ms
History
☐ Enable storage to historian
☐ Enable Tag Hierarchy

FIGURE 8: "RESTART THE ENGINE WHEN IT FAILS" OPTION ELIMINATED IN WAS 3.5

Eliminated "Restart Engine Process When Transitioning from Active to Standby"

This option is eliminated in the AppEngine's Redundancy Configuration (Figure 9 below). In WAS 3.5, the AppEngine is always restarted when it switches from Active to Standby. In WAS 3.1 or before, many failover stability issues were linked to having this option unchecked.



Before:

Maximum consecutive heartbeats missed from Active engine:
Maximum consecutive heartbeats missed from Standby engine:
Maximum time to maintain good quality after failure:
Maximum time to discover partner:
Force failover command:
☑ Restart engine process when transitioning from Active to Standby

After:

Maximum consecutive heartbeats missed from Active engine: 5
Maximum consecutive heartbeats missed from Standby engine: 5
Maximum time to maintain good quality after failure: 15000 ms
Maximum time to discover partner: 15000 ms
Force failover command:

Figure 9: "Restart engine process when transitioning from Active to Standby" Option is Eliminated

Increased "WatchdogStartupTimeout" and "WatchdogShutdownTimeout" Threshold Settings

This setting value is increased from **30,000** ms (30 seconds) to **300,000** ms (5 minutes).

Many redundancy performance issues found in WAS 3.1 (or before) were related to the AppEngine being prematurely killed, because the startup or shutdown timeout values were too short. As a result, the AppEngine was allowed to reach and stay in illegal transitional states (Failed to Start, Failed to Shudown) etc..

Using the higher settings can prevent this issue from occurring.

C. He

Back to top