Tech Note 401
Fine-Tuning AppEngine Redundancy Settings

Topic#: 002079
Created: February 2005
Updated: December 2005

Introduction

This technote provides the guidelines to establish settings for redundancy-configurable parameters based on size of the application. In addition, this document provides tips related to redundancy-enabled engine deployment.

Redundancy Settings

Figure 1 (below) shows the default settings for AppEngine Redundancy listed in the **Redundancy** tab field:

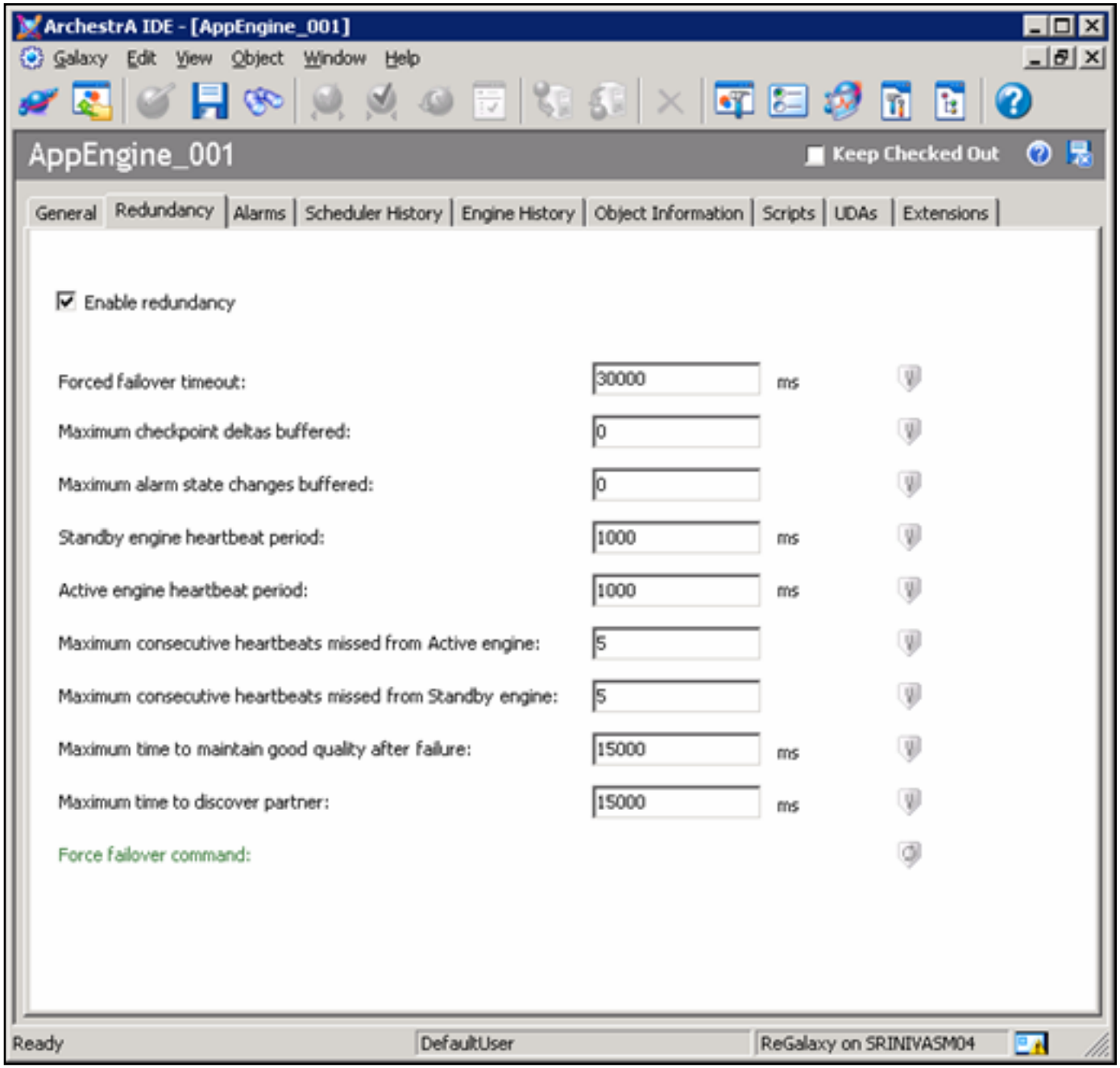

Figure 1: Default Redundancy Settings

The above settings are good for standard application size of 3-5,000 I/O (five hundred AppObjects with 10 I/O per Object).

When the application size increases, the above settings must be changed (tuned) to avoid unexpected behavior during maintenance and under normal operating conditions of the production system.

It is very important to understand the redundancy-configurable parameters before you change them. The following information explains the details of the parameters and their settings. Once the redundancy settings are changed, run test cycles to ensure that the settings are appropriate for the size of your application.

1. **Forced failover timeout**: The maximum allowed time (milliseconds) for a standby engine to become active and execute all objects hosted to it for a single scan. Assuming the force failover has been initiated via ForceFailoverCmd. If the standby engine does not become active within this time period, this engine will revert back to the active engine. The Default setting is 30,000 ms. This setting is recommended for up to 5,000 I/O points-per redundancy-enabled Engine.

   Wonderware® QA tests have proven that for 40,000 I/O with 5000 AppObjects (Eight I/O-per-Object) took 3 minutes and 33 seconds to failover. It is safe to set **45,000** ms. between 3-5,000 I/O points, **180,000** ms up to 20,000 I/O and **240,000** ms from 20,000 to 40,000 I/O.

   300,000 ms. is recommended for systems with more than 40,000 I/O points.

2. **Maximum checkpoint deltas buffered**: Maximum number of checkpoint deltas that can be buffered before a full checkpoint synchronization is performed. In the case of Checkpoint deltas being sent to the standby engine so quickly they cannot be processed fast enough by the standby engine, checkpoints will be buffered.

   The default setting is **0**. This setting has no negative impact, and tuning is not recommended.

3. **Maximum alarm state changes buffered**: Maximum number of alarm state changes that can be buffered before a full snapshot of the engines' alarm state changes is performed. In the case of Alarm state changes being sent to the standby engine so quickly they cannot be processed fast enough by the standby engine, alarm state changes will be buffered.

   The default setting is **0**. This setting has no negative impact, and tuning is not recommended.

4. **Standby engine heartbeat period**: Time interval (milliseconds) at which heartbeats are sent by the failover service on the standby engine to the failover service on the active engine over the RMC.

   The default setting is **1000** ms. Tuning is not recommended.

5. **Active engine heatbeat period**: Time interval (milliseconds) at which heartbeats are sent by the failover

service on the active engine to the failover service on the standby engine over the RMC.

The default setting is **1000** ms. Tuning is not recommended.

6. **Maximum consecutive heartbeats missed from Active engine**: The number of heartbeats from the Active engine that can be missed before a bad connection (over the RMC with the standby engine) is assumed by the Standby engine.

   The default setting is **5**. This means is that if the Standby engine doesn't receive any heartbeats within 5 seconds from the Active engine, the standby will assume that he cannot reach the Active engine via RMC.

   In a multiple redundant engine pair environment, set this value to a larger number to avoid false failover(s) when the system is busy. The recommended setting is **10-30** seconds for systems with 5-40,000 I/O points. In a very large system environment (40,000 + I/O points) **~60** is recommended.

7. **Maximum consecutive heartbeats missed from Standby engine**: The number of heartbeats from the Standby engine that can be missed before a bad connection (over the RMC with the standby engine) is assumed by the Active engine. When a bad connection is detected the Active engine will switch to Active-standby Not Available status.

   The default setting is **5**. This means that if the active engine doesn't receive any heartbeats within 5 seconds from the Standby engine, the Active engine will assume that he cannot reach the standby node via RMC.

   In a multiple redundant engine pair environment, set this value to a larger number. The recommended setting is **10-30** seconds for systems with 5-40,000 I/O points. In a very large system environment (40,000 + I/O points) **~60** is recommended.

8. **Maximum time to maintain good quality after failure**: Determine how long to wait after standby engine has become Active and binds subscribed references to this engine before the quality can be set to uncertain. The default setting is **15,000** ms. For systems between 5-40,000 I/O points, **120,000** ms. is recommended. For systems with more than 40,000 I/O points, **150,000** ms. is recommended.

> **Note:** Assuming remote I/O, setting the value too low causes all I/O references to unsubscribe, then re-subscribe on failover. The optimum setting ensures that remote I/O references are preserved for failover. This behavior also applies in the RDI Object context.

9. **Maximum time to discover partner**: How long to wait when attempting to connect to the failover partner before setting the failover partner state to **Unknown**. The default setting is **15,000** ms. Tuning is not required for this setting.

10. **Restart the engine when it fails**: This feature is not recommended to use in a redundancy environment. If the engine hangs or fails, failover occurs. By default this check box is not selected (Figure 2 below):

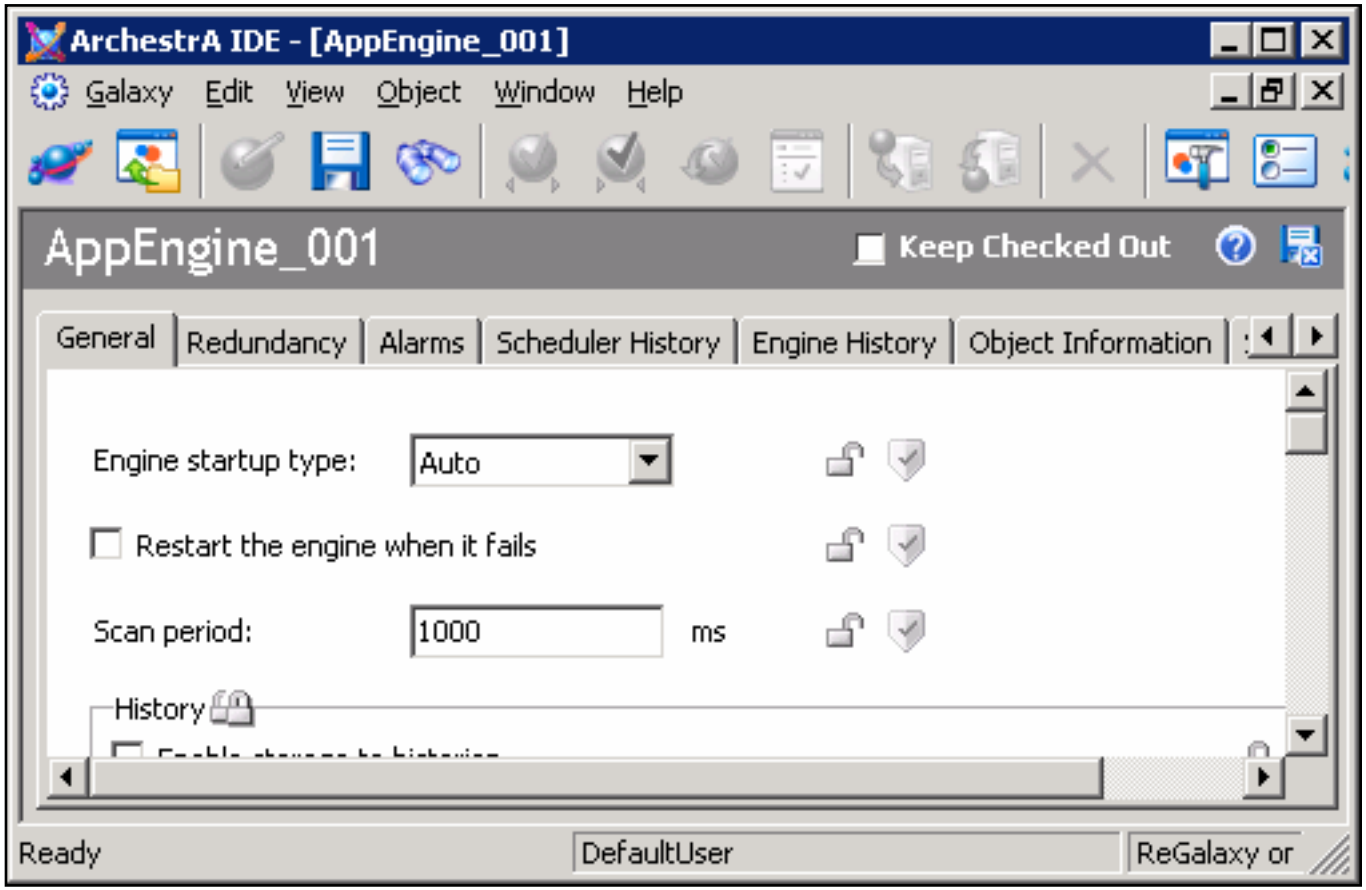

Figure 2: AppEngine Editor

11. **Checkpoint period**: Set to **0** by default. Checkpoints will be written to the disk at the engine scan period intervals.

    Do not set this value to too low or too high. Writing to disk at every scan period is resource-intensive. For a 20,000 I/O application, **20,000** ms. is recommended. For systems with 40,000 I/O points or more, **60,000** ms. is recommended.

It is important to note that in a redundancy environment, checkpoint synchronization is performed at every scan and the synchronized data is stored in the memory on both Active and Standby nodes. If the Standby node has to become active, it will first use this data for startup.

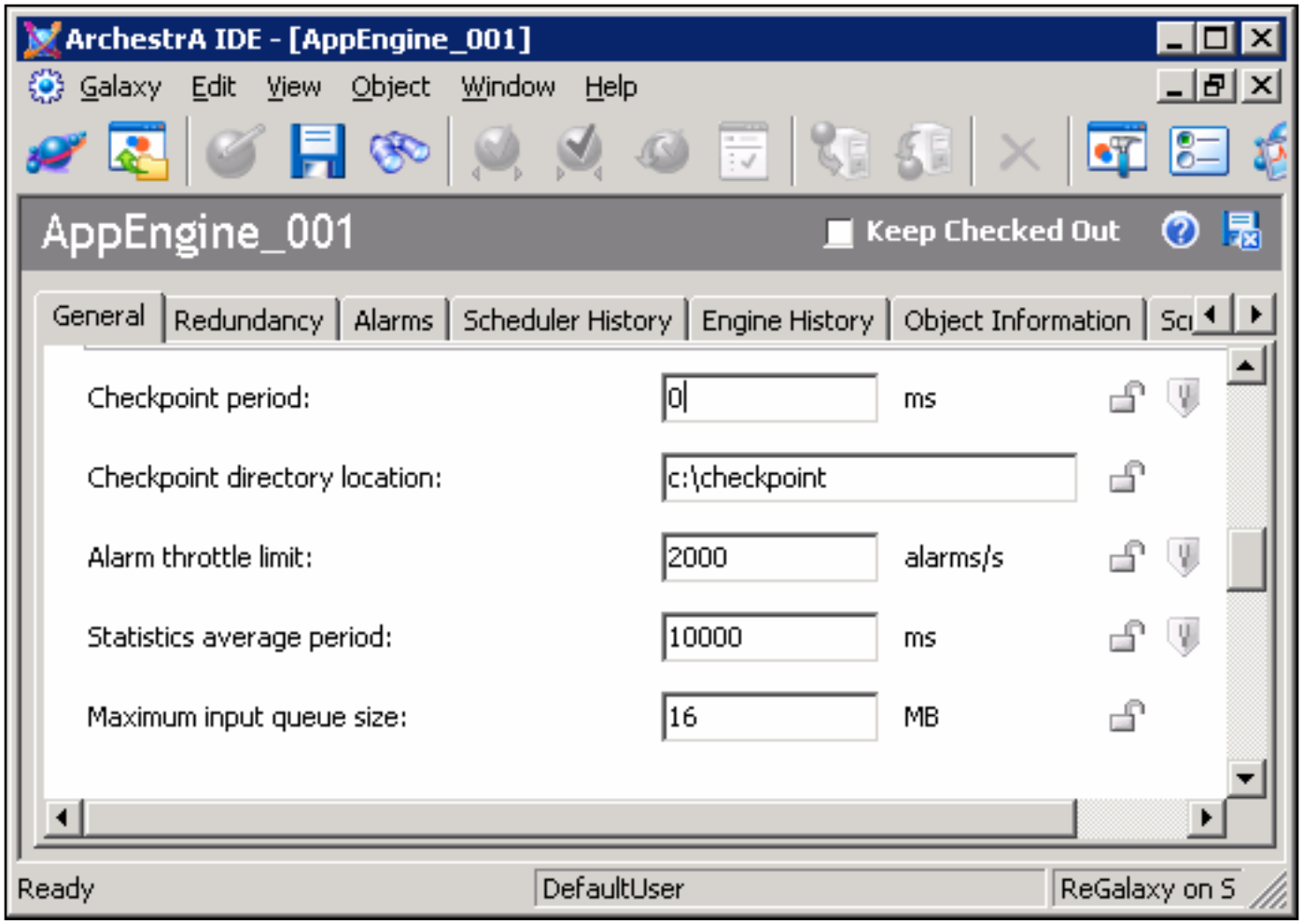

Figure 3: Checkpoint Period Default

12. **Engine Failure Timeout**: The amount of time (milliseconds) that the engine can go without notifying the Bootstrap of its health. After this time, the EngineFailureAlarm is set to **TRUE** in the Engine.

    Ensure that the value set into this attribute is not less then the engine's configured scan period. The default value is set to **10,000** ms as shown in Figure 4 (below).

Internally, the system will triple the value that you specify for this attribute. This means that the engine can go without notifying the bootstrap of its health for 30 seconds if the value is set to 10 seconds. The failover will then occur after 30 seconds.

In a redundancy setup the recommendation is to set this value to **5,000** ms.

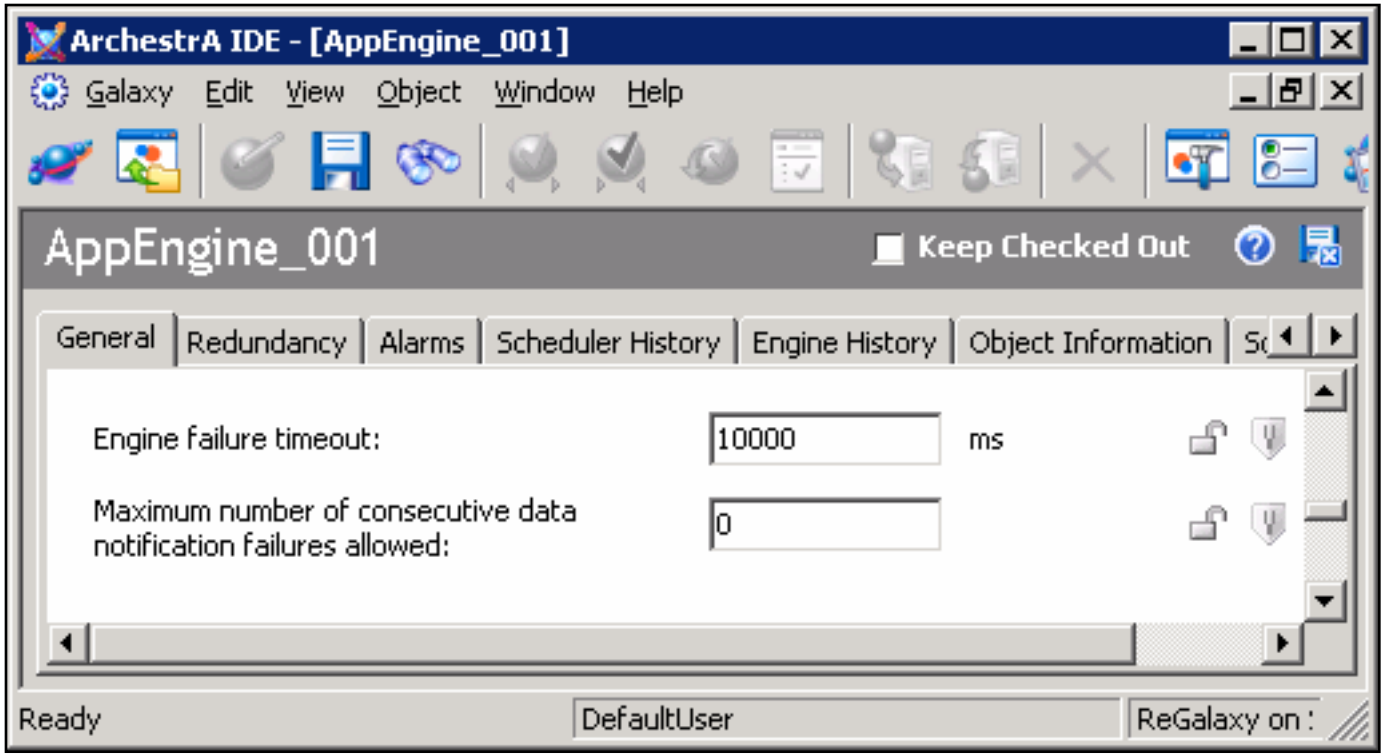

Figure 4: Engine failure timeout interval

Platform Settings

1. **Message Timeout**: The number of milliseconds for response message MxTimeouts for cross-engine communications. The default value is 30,000 ms as shown in **Figure 5** (below).

   It is recommended to set this at a higher number on GR Node to avoid engine communications issues during deployment of large Galaxies. The recommended setting is **300,000** ms.

2. **NMX Heartbeat period**: The frequency in milliseconds at which heartbeats are sent to other platforms. Heartbeats will only be set up between platforms when one platform needs to know if the other is still

running.

For example, if an engine on platform **X** is subscribed to data on an engine deployed to platform **Y**, the heartbeats will be sent by platform **Y** to platform **X**. The default value is **2,000** ms. as shown in platform editor. No tuning is recommended for this setting.

3. **Consecutive number of missed NMX Heartbeats allowed**: Maximum allowed number of consecutive heartbeats missed prior to failing the primary communications channel to this platform. The default value is **3** as shown in platform editor.

   This means that if an engine on platform **X** is subscribed to data on an engine deployed to platform **Y**, heartbeats will be sent by Platform **Y** to platform **X**. If platform **X** doesn't receive any hearbeats within 6 seconds from platform **Y**, Platform **X** assumes that communication is lost to platform **Y**.

   It is recommended this value is set to **6** on the GRNode to avoid communication issues during deployment of a large Galaxy (20-40,000 or more I/O points).

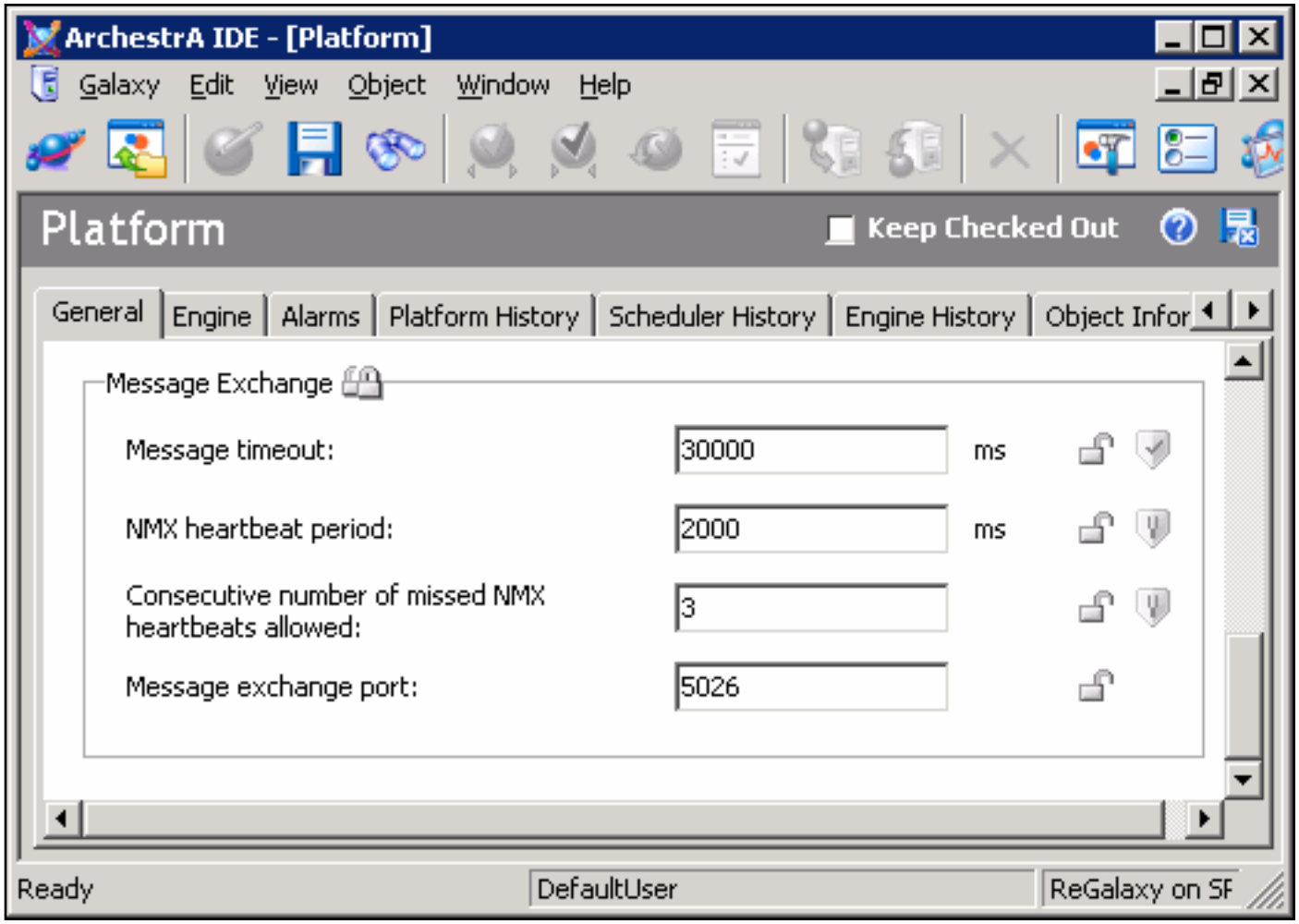

Figure 5: Missed NMX Heartbeats

4. **Deployment**: For small galaxies, a Cascade deployment from the **Galaxy Object** level is optimal.

   For large galaxies, perform Cascade deployment from the **Platform** Level. When performing Cascade Deployment from the Platform Level, it is important to make sure that you start deploying first with platforms hosting primary engine(s) and then deploy platforms hosting backup engine(s). By doing this you are starting primary as active and all objects running under this engine.

   If you deploy platforms hosting backup engine first, this will become active first and then when you deploy platform with primary engine, all objects will be running to active engine causing issues related to load-balancing during deployment.

S. Mariyala

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 **www.wonderware.com/support/mmi**

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

 **back to top**