OMNIbus Heartbeat Solution

Author: Nick Lansdowne **Date: 14th December 2010**

Introduction

The OMNIbus Heartbeat Solution is designed to monitor nodes, applications or application instances through the use of heartbeat pulse events, i.e. events received at a regular interval from the monitored resource. The solution is designed to process such pulse heartbeat pulse events and generate a “missed heartbeat” alert if a pulse isn’t received in a timely fashion. Each pulse instance can be assigned a unique pulse interval and “missed heartbeat” severity, or use centrally assigned defaults. A number of additional configuration options are available for tailoring the solution.

The solution correlates missed heartbeats, suppressing application and application /instance alerts where the owning node has missed a heartbeat, and similarly suppressing application instances alerts where the owning application has missed a heartbeat.

The following sections document the installation, removal, configuration and use of the solution.

Note: This solution is offered as-is and as such is provided with no support from Orb Data Limited.

Installation Steps

The solution is based on OMNIbus Probe rules, ObjectServer triggers and a WebGUI tool.

The solution consists of a number of files to be imported into the ObjectServer, Probes or WebGUI server. All files are located on the Orb Data CVS system: http://customer.orb-data.com/OrbData/OMNIbus/.

File	Server	Comments
orbHeartbeat.updates.sql	ObjectServer	Create the ObjectServer objects required for the heartbeat processing, including a custom database, procedures, triggers, signals and roles required for the processing of the Heartbeat events.
orbHeartbeat.removal.sql	ObjectServer	Removes the objects created by the updates script
orbHeartbeat.bigate.map	Bi-directional Gateway	Custom heartbeat table mapping entries for the bi-directional gateway
orbHeartbeat.bigate.tblrep	Bi-directional Gateway	Custom heartbeat table replication definitions for the bi-directional gateway

orbHeartbeat.include	EIF Probe	EIF Probe logix=c for processing the EIF Heartbeat events
orbHeartbeat.targets.include	EIF Probe	Defines the target ObjectServers and the customer Heartbeat table
orbHeartbeat.config.include	EIF Probe	User configuration properties that are applied by the heartbeat processing logic
orbHeartbeat.tools.xml	WebGUI	AEL tool for decommissioning a heartbeat

Step 1: ObjectServer Update

Create the ObjectServer custom database, procedures, triggers etc using the script orbHeartbeat.updates.sql. Complete the following steps on relevant ObjectServers, including any Backup ObjectServer if running as part of a virtual pair:

1. Copy the file orbHeartbeat.updates.sql to the ObjectServer
2. Run the SQL script against the ObjectServer from a terminal session, using nco_sql:

   ```
   nco_sql –server <OS> -user <user> -password <pwd> < orbHeartbeat.updates.sql
   ```

   where <OS> is the ObjectServer alaias (as defined in the interfaces file) and <user> is a valid ObjectServer system user.
3. On the Backup ObjectServer only, disable the Trigger group orb_heartbeat

No errors should be reported by the above script, unless updating a Backup ObjectServer where roles and groups have been replicated from the Primary ObjectServer.

The heartbeat triggers on the backup ObjectServer should be enabled/disabled automatically during a fail-over/fail-back. This can be achieved by either moving all triggerd into the *primary_only* trigger group or updating the procedures *enable_automation* and *disable_automation* to additionally toggle the state of the *orb_heartbeat* trigger group.

Step 2: ObjectServer Bi-directional Gateway

If running a virtual pair of ObjectServers, synchronised by a bi-directional gateway, then it will be necessary to update the mapping and table synchronisation files to synchronise the custom heartbeat database.

To update the Bi-directional gateway configuration:

1. Edit the active bi-directional gateway mapping file, adding the mapping contained within the file orbHeartbeat.*bi-gate.map*
2. Edit the active bi-directional gateway table replication file, adding the entries in the file orbHertbeat.*bi-gate.tbldef*

Step 3: WebGUI

Create the WebGUI Tool for decommissioning the heartbeats.

1. Copy the file orbHeartbeat.tools.xml to the WebGUI Server
2. From a terminal session, change into the *waapi* binary directory

   ```
   cd $TIPHOME/products/ncw/waapi/bin/
   ```

3. Load the tool:

   ```
   runwaapi –user <tipadmin> -password <pwd> -file /<PATH>/
   orbHeartbeat.tools.xml
   ```

   where <tipadmin> is a TIP Administrator user,

4. Confirm no errors are reported by the above script

Step 4: EIF Probe

The supplied rules files are designed to be loaded into the structure default Tivoli EIF Rules. The following steps are required:

1. Copy the three include files onto the EIF Probe Server
2. Set the environment variable ORB_HEARTBEAT_RULES to the path containing the heartbeat include files. This may be required in the profile for the user starting the EIF Probe or the Process Control start-up script, if the probe is being started by Process ControlThe following syntax can be used for most UNIX/Linux platforms:

   export ORB_HEARTBEAT_RULES=<PATH>

3. Update the EIF Rules file, adding an include statement to the beginning of the file for the targets file:

   include “$ORB_HEARTBEAT_RULES/orbHeartbeat.targets.include”

4. Update the body of the EIF Rules file, adding an include statement with the main switch statement for the source:

   include “$ORB_HEARTBEAT_RULES/orbHeartbeat.include”

5. Update the file $ORB_HEARTBEAT_RULE/orbHeartbeat.targets.include, using a standard text editor. Edit the two “registertarget” statements to reference the ObjectServer(s). The first entry in quotes for each line should match the “Server” entry in the current probe properties file. The second entry the “ServerBackup” entry. For example:

Properties File Entry	Register Target Statement	Comment
Server: “NCOMS” ServerBackup: “”	StandardAlerts = registertarget("NCOMS", "", "alerts.status", "alerts.details") OrbHeartbeat = registertarget("NCOMS", "", "custom.orbHeartbeat", "alerts.details")	Single ObjectServer defined

Server: “NCOMS_P” ServerBackup: “NCOMS_B”	StandardAlerts = registertarget("NCOMS_P", " NCOMS_B ", "alerts.status", "alerts.details") OrbHeartbeat = registertarget("NCOMS_P", " NCOMS_B", "custom.orbHeartbeat", "alerts.details")	Primary and Backup separately references
Server: “NCOMS_V” ServerBackup: “”	StandardAlerts = registertarget("NCOMS_V", "", "alerts.status", "alerts.details") OrbHeartbeat = registertarget("NCOMS", "", "custom.orbHeartbeat", "alerts.details")	Virtual ObjectServer defined

6. The probe will need to be restarted or the process “HUPed”

Configuration

Configuration options are available for the EIF Probe Rules, the ObjectServer triggers. Additionally, the WebGUI Tool must be assigned to relevant users.

ObjectServer Configuration

Configuration of the ObjectServer is completed by the procedure labelled *orbHeartbeat_GetConfig*. This procedure may be updated from the ObjectServer administrator. The following options are available.

Property	Default	Comments
AlertGroup	'Orb Heartbeat' (String)	The *AlertGroup* assigned to all Orb Data Heartbeat related events. Permitted values: Unique string
Class	86000 (numeric)	The Class ID assigned to all Orb Data Heartbeat related events. If this value is changed a conversion must exist for the new value. Permitted values: Integer, as defined in the Class Conversions table
enableLog	1	Not currently used
defaultSeverity	4 (Major)	Severity assigned to a Missed Heartbeat alert if the slot “hbSeverity” is not defined in the received Heartbeat Pulse. Permitted values: 1 to 5 (or as defined in the severity conversions table)
defaultInterval	300 (seconds)	Heartbeat pulse interval used if the slot *hbInterval* is not defined in the received heartbeat pulse event. Permitted values: Integer > 0

Property	Default	Comments
defaultLeniency	0 (percentage)	Heartbeat pulse leniency used if the slot *hbLeniency* is not defined in the received heartbeat pulse event. This percentage is added to the interval to define the time before a missed heartbeat alert is generated. Permitted values: 0-100%
newHeartbeatSeverity	1 (Intermediate)	The severity assigned to the alert generated when a heartbeat pulse is received for a new node/application/instance. Set this to 0 to disable this alert. Permitted values: 0 to 5 (or as defined in the severity conversions table)
expireNewHeartbeat	300 (seconds)	The expire time assigned to the alerts generated for new heartbeat pulses events. The event will be deleted after this time (assuming the IBM shipped *expire* trigger is enabled) Permitted values: Integers >= 0
sendEventEachCycle	0 (No)	Enables/disables the sending of missed heartbeat events at every cycle (i.e. every time the analysis trigger runs). Permitted values: 0=No, 1=Yes

EIF Probe Configuration

The EIF Probe configuration is completed in the rules file *orbHeartbeat.confg.include*. This file can be edited in a standard text editor. The probe will need to be restarted or “HUPed” after any changed.

Property	Default	Comments
$orbHeartbeat_DiscardInvalidEvent	false (Boolean)	The hostname slot must be set within all heartbeat events received by the probe. If an event is received without a hostname slot defined, an operator alert can be generated, dependent on this configuration value. Permitted values: false/true
$orbHeartbeat_InvalidEventSeverity	4 (Major)	The severity assigned to any invalid heartbeat events. Permitted values: 1 to 5 (or as defined in the severity conversions table)

Note: The *orbHeartbeat.targets.*include file must be updated as part of the solution installation.

WebGUI Configuration

The WebGUI alerts tool named *orbHeartbeat_decomission* is created by the XML file. This tool must be added to the *Alerts* menu by a WebGUI administrator. The administrator will likely wish to assign a label to the tool once assigned to the alerts menu. It is the label that is displayed on the Alerts menu when accessed from the Active Event List.

To assign the tool the alerts menu, log-in to the WebGUI using credentials assigned the *ncw_admin* role, and select the navigator menu *Administration→Event Management Tools→Menu Configuration*. Modify the *alerts* menu to add the *orbHeartbeat_decommission* tool.

By default users within the ObjectServer *Administrators* group are granted access to the tool. Additionally, ObjectServer users can be assigned to the group *orbHeartbeat* to grant access to the tool.

Use of the Solution

Heartbeat Pulse Events

EIF “heartbeat pulse events” generated by the nodes, applications or application instances to be monitored must populate the attributes (slots) indicated in the table below.

Attribute	Value	Comment
Class	orbHeartbeatPulse	Mandatory
source	Orb Data Heartbeat	Mandatory
hostname	Monitored system name	Mandatory
Application	Monitored application name	Optional
Instance	Monitored application instance	Optional
hbInterval	Integer (seconds)	Time interval of heartbeat pulses. Optional (default assigned centrally if not populated) For example, hbInterval=300 (5 minutes)

hbLeniency	Integer (percentage)	Additional time, as a percentage of the hbInterval, above the hbInterval before a missed heartbeat event is generated. (default assigned centrally if not populated) For example: hbInterval=300 hbLeniency=10 Missed heartbeat generated after 330 seconds (5 ½ minutes)
hbSeverity	Integer (1-5)	Severity assigned to a missed heartbeat alert. Optional (default assigned centrally if not populated).

Decommissioning Heartbeat Monitored Resource using the WebGUI

If a Node, Application or Application Instance that has been generating heartbeat pulses is decommissioned, then the ObjectServer will generate missed heartbeat alerts for the resource. The WebGUI Tool *orbHeartbeat_decommission* can be used to suppress alerts for the resource, and any child resources.

From the Active event List, right click on the missed heartbeat alert for the decommissioned resource, and select the heartbeat decommission tool (note the name of the tool will be assigned by the WebGUI administrator during installation).

Decommissioning Heartbeat Monitored Resource using an EIF Event

Decommissioning of a resource can be achieved by generating an EIF Event. The attributes within the following table are required.

Attribute	Value	Comment
Class	orbHeartbeatStop	Mandatory
source	Orb Data Heartbeat	Mandatory
hostname	Monitored system name	Mandatory
Application	Monitored application name	Optional
Instance	Monitored application instance	Optional