The vSphere High Availability (HA) feature for ESX and ESXi hosts in a cluster provides protection for the Guest OS and applications running in a virtual machine by restarting the virtual machine if an OS or application failure occurs.

The High Availability feature provides this reset capability through two different mechanisms:

- Guest OS heartbeat issued by the VMware Tools process, known as VM Monitoring
- Heartbeat issued by a developer’s script or application that uses the High Availability Application Monitoring SDK to communicate with the VMware Tools process and the vSphere High Availability agent, known as Application Monitoring. In this option, the processes monitor locally, which avoids the overhead of sending messages to and from the vCenter server.

This technical note discusses only Application Monitoring. This document includes the following topics:

- “Before You Begin” on page 2
- “Using the High Availability Application Monitoring APIs” on page 2
- “High Availability Application Monitoring API Error Messages” on page 5
With the High Availability Application Monitoring SDK, developers can create application monitoring programs. The application monitoring program sends an enable request to start the monitoring, followed by a heartbeat signal. The vSphere infrastructure passes the signal up from your application monitoring program to the virtual machine, and then to the ESX or ESXi host. The High Availability monitoring agent will reset the virtual machine when the application monitoring program stops sending a heartbeat signal.

**Figure 1. Heartbeat and Status Signals**

**Before You Begin**

Before you begin working with the High Availability Application Monitoring SDK, make sure that your vSphere application is running within a VMware ESX or ESXi cluster that has the High Availability and VM and Application Monitoring options enabled.

VMware's New Cluster Wizard allows you to choose from three monitoring options: Disabled, VM Monitoring Only, and VM and Application Monitoring. If you choose the VM Monitoring Only option, you will have the Guest OS monitoring discussed previously, but you will not have the ability to use the High Availability Application Monitoring SDK, which is the Application Monitoring part of the VM and Application Monitoring option.

You must also install VM Tools on the virtual machines where your monitoring applications are running.

**Using the High Availability Application Monitoring APIs**

You can use the High Availability Application Monitoring SDK to create a stand-alone application monitoring program, or to enhance an existing application or script. The purpose of your application monitoring program will determine the API call sequence and the application behavior that you write to handle the response data.

For example, if your application monitoring program is tracking other critical applications or processes that are running within the virtual machine, you might want to intentionally stop sending the heartbeat signals if one of these processes fails. The High Availability monitoring agent interprets the absence of a heartbeat as a failure and resets the virtual machine.

Most of the calls you make using the High Availability Application Monitoring APIs send information one-way to the virtual infrastructure of the ESX or ESXi host, and the host relays the information to the High Availability monitoring agent. The GetAppStatus call, however, is a two-way transaction that lets you request the virtual machine status from the High Availability monitoring agent.

These calls do not have input parameters, because all of the calls are local. The vSphere infrastructure passes the heartbeat and status data to and from the other levels of the cluster.
Call each function from your application monitoring program. The virtual infrastructure (in the virtual machine where the application monitoring program is running) passes the function data up to the ESX or ESXi host. The local virtual machine sends all status responses to your application monitoring program, even though they are passed down from the High Availability monitoring agent.

**High Availability Application Monitoring API Functions**

You can include the following monitoring calls within a vSphere application monitoring program:

**Table 1. High Availability Application Monitoring API Calls**

<table>
<thead>
<tr>
<th>Call Name</th>
<th>Data Type Returned</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMGuestAppMonitor_Enable</td>
<td>char</td>
<td>Requests that the virtual machine infrastructure monitors the calling application. The virtual machine infrastructure returns a value of VMGUESTAPPMONITORLIB_ERROR_SUCCESS, if monitoring is enabled. After your application monitoring program makes this call, your program must call VMGuestAppMonitor_MarkActive() at least once every 30 seconds or the virtual machine infrastructure will change the virtual machine's status to Red or Gray.</td>
</tr>
<tr>
<td>VMGuestAppMonitor_Disable</td>
<td>int</td>
<td>Requests that the virtual machine infrastructure stop monitoring the calling program. The virtual machine infrastructure returns a value of TRUE, if monitoring is disabled.</td>
</tr>
<tr>
<td>VMGuestAppMonitor_IsEnabled</td>
<td>int</td>
<td>Returns the current recorded state of application monitoring. The virtual machine infrastructure returns a value of TRUE, if monitoring is enabled.</td>
</tr>
<tr>
<td>VMGuestAppMonitor_MarkActive</td>
<td>char</td>
<td>Sends a request to mark the program as active. This function is also known as the heartbeat because your program must call it at least once every 30 seconds while the application monitoring is enabled, or the virtual machine infrastructure will determine that the monitoring has failed.</td>
</tr>
</tbody>
</table>
Calling the APIs from Your Application

The following steps provide a possible API sequence of calls:

1. Include `vmGuestAppMonitorLib.h` in the declarations for your C program.

2. To start the monitoring, notify the virtual machine that you are going to start sending a heartbeat signal by calling `VMGuestAppMonitor_Enable`.

3. After you have called `VMGuestAppMonitor_Enable`, call `VMGuestAppMonitor_MarkActive` every 30 seconds or your virtual machine will be reset.

4. Send `VMGuestAppMonitor_IsEnabled` to make sure the virtual machine infrastructure received your requests correctly and has begun monitoring.

5. Periodically, call `VMGuestAppMonitor_GetAppStatus` to make sure the vSphere infrastructure is still receiving the heartbeat calls.

The status will be returned as `Green`, `Red`, or `Gray`. See "High Availability Application Monitoring API Calls," for a description of each status value. The following diagram shows a possible coding flow for the `GetAppStatus` call.
6 After you call VMGuestAppMonitor_GetAppStatus, call the VMGuestAppMonitor_Free function to free the memory that was used to store the status.

If your application does not free the memory, it can use a large amount of storage very quickly, because a new status is created every 30 seconds, when VMGuestAppMonitor_MarkActive is called.

7 Call VMGuestAppMonitor_Disable when you want the agent to stop monitoring.

**appmon.cpp Code Sample**

The High Availability Application Monitoring SDK includes a code sample called appmon.cpp. The sample is located in the /docs/samples directory and defines the entry point for the console application. appmon.cpp includes interface code that your application monitoring program can send after receiving results from the VMGuestAppMonitor_Enable, VMGuestAppMonitor_MarkActive, and VMGuestAppMonitor_Disable calls.

**High Availability Application Monitoring API Error Messages**

The infrastructure can return the following errors in response to the High Availability Application Monitoring SDK calls.

<table>
<thead>
<tr>
<th>Error Message</th>
<th>Data Type</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMGUIDSTAPPMONITORLIB_ERROR_SUCCESS</td>
<td>int</td>
<td>0</td>
<td>Call completed successfully.</td>
</tr>
<tr>
<td>VMGUIDSTAPPMONITORLIB_ERROR_OTHER</td>
<td>char</td>
<td>0</td>
<td>Unknown error.</td>
</tr>
<tr>
<td>VMGUIDSTAPPMONITORLIB_ERROR_NOT_RUNNING_IN_VM</td>
<td>char</td>
<td>0</td>
<td>Calling application is not running within a virtual machine.</td>
</tr>
<tr>
<td>VMGUIDSTAPPMONITORLIB_ERROR_NOT_ENABLED</td>
<td>char</td>
<td>0</td>
<td>Monitoring is not enabled.</td>
</tr>
<tr>
<td>VMGUIDSTAPPMONITORLIB_ERROR_NOT_SUPPORTED</td>
<td>char</td>
<td>0</td>
<td>Monitoring is not supported.</td>
</tr>
</tbody>
</table>