vSphere Guest Programming Guide

VMware vSphere Guest SDK 4.0

EN-000138-00

You can find the most up-to-date technical documentation on the VMware Web site at:

http://www.vmware.com/support/

The VMware Web site also provides the latest product updates.

If you have comments about this documentation, submit your feedback to:

docfeedback@vmware.com

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

About This Book

The *vSphere Guest Programming Guide* provides information about developing applications using the VMware® Guest Application Programming Interface (API). VMware provides several different software development kit (SDK) products, each of which targets different developer communities and target platforms. This guide is intended for developers who want to retrieve information about the virtual machine and host hardware in which the application runs. The supported platforms include VMware ESX 3.0.x, VMware ESX/ESXi 3.5, and VMware ESX/ESXi 4.0.

Revision History

This book is revised with each release of the product or when necessary. A revised version can contain minor or major changes. Table 1 summarizes the significant changes in each version of this book.

Table 1. Revision History

Revision Date	Description
18JUL2005	Initial release of the VMware Guest SDK providing support for VMware ESX 3.0.
29NOV2007	No new information, but revised to note support for VMware ESX 3.5 and VMware ESX 3i version 3.5.
7MAY2009	Revised manual for VMware ESX version 4.0.

Intended Audience

This book is intended for anyone developing applications to run inside guest operating systems under VMware ESX.

Document Feedback

VMware welcomes your suggestions for improving our documentation. Send your feedback to docfeedback@vmware.com.

Technical Support and Education Resources

The following sections describe the technical support resources available to you. To access the current versions of other VMware books, go to http://www.vmware.com/support/pubs.

Online and Telephone Support

To use online support to submit technical support requests, view your product and contract information, and register your products, go to http://www.vmware.com/support.

Support Offerings

To find out how VMware support offerings can help meet your business needs, go to http://www.vmware.com/support/services.

VMware Professional Services

VMware Education Services courses offer extensive hands-on labs, case study examples, and course materials designed to be used as on-the-job reference tools. Courses are available on-site, in the classroom, and live online. For on-site pilot programs and implementation best practices, VMware Consulting Services provides offerings to help you assess, plan, build, and manage your virtual environment. To access information about education classes, certification programs, and consulting services, go to http://www.vmware.com/services.

VMware vSphere Guest SDK Programming Guide

The VMware Guest API provides functions that you can use in a program that runs in the guest operating system environment on a VMware ESX virtual machine. This guide includes the following topics:

Overview of the vSphere Guest API

The vSphere Guest API provides functions that management agents and other software can use to collect data about the state and performance of a VMware ESX virtual machine. The API provides fast access to resource management information, without the need for authentication.

The Guest API provides read-only access. You can read data using the API, but you cannot send control commands. To issue control commands, use the vSphere Web Services SDK. For details, see the *VMware vSphere Web Services SDK Programming Guide* and the *VMware vSphere API Reference*, which are available on the VMware Web site.

Supported Guest Operating Systems

The vSphere Guest API software runs on most Windows or Linux guest operating systems supported by VMware ESX 3.0.x, VMware ESX/ESXi 3.5, or VMware ESX/ESXi 4.0. The Guest API does not support the following guest operating system environments:

- Windows 9x.
- WinNT4. To use the Guest API on WinNT4 you must use Guest SDK 3.5; go to http://www.vmware.com/support/developer/guest-sdk.

See the *Guest Operating System Installation Guide* for a list of supported guest operating system versions.

Virtual Machine Statistics

With the vSphere Guest API, you can monitor various statistics about the virtual machine. You can use this information to retrieve scheduling and resource usage information about the environment. With these statistics, the virtual machine can react to changes in the virtual environment immediately at the application layer.

The following list shows some of the information that you can retrieve through the vSphere Guest API:

- Amount of memory reserved for the virtual machine.
- Amount of memory being used by the virtual machine.
- Upper limit of memory available to the virtual machine.
- Number of memory shares assigned to the virtual machine.

- Maximum speed to which the virtual machine's CPU is limited.
- Reserved rate at which the virtual machine is allowed to execute. An idling virtual machine might consume CPU cycles at a much lower rate.
- Number of CPU shares assigned to the virtual machine.
- Elapsed time since the virtual machine was last powered on or reset.
- CPU time consumed by a particular virtual machine. When combined with other measurements, you can estimate how fast the virtual machine's CPUs are running compared to the host CPUs.

IMPORTANT The API uses a handle that provides access to the statistics. The handle also is a mechanism to determine whether the API can provide accurate information. (Certain events, such a migrating a virtual machine with VMotion™, temporarily make it impossible to provide accurate information.)

How to Use the vSphere Guest API

The vSphere Guest API defines functions and data types that you use to extract virtual machine data. This section covers the following topics:

vSphere Guest API Runtime Components

To use the vSphere Guest API, the runtime components must be installed in the guest operating system. The runtime components are dynamically loaded binary modules for 32-bit and 64-bit guests. When you install VMware Tools, the vSphere Guest API runtime components are installed automatically. You can also download them from http://www.vmware.com/download/sdk/guest_sdk.html.

To make the vSphere Guest API functions available to your program, use your program's standard methods to load the library.

- In a Windows guest operating system, the library file is `vmGuestLib.dll`. The import library file is `vmGuestLib.lib`.
- In a Linux guest operating system, the library file is `libvmGuestLib.so`.

IMPORTANT If you are using a Security-Enhanced Linux (SELinux) guest operating system, the security policies might interfere with dynamic loading of `libvmGuestLib.so`. Refer to documentation about SELinux policy configuration.

The vSphere Guest SDK includes the test program `vmGuestlibTest.c`. If you are using a Windows environment, you must rebuild the test program. The `vmGuestLib.dll` library file is a non-Unicode DLL. In Microsoft Visual Studio, build the test program `vmGuestlibTest.c` as a non-Unicode executable so that the program can access the DLL at runtime.

Enabling and Disabling the Runtime Components

The vSphere Guest API runtime components are enabled by default (`disable = "FALSE"`). To disable the runtime components, use the configuration editor in the vSphere Client to edit the configuration file for the virtual machine. The virtual machine must be powered off before you can use the configuration editor.

1 In the vSphere Client window, right-click the virtual machine in the machine list.

2 In the drop-down menu, select **Edit Settings**.

3 In the Virtual Machine Properties window, click the **Options** tab.

4 In the list of "Advanced" settings, select **General**.

5 Click **Configuration Parameters**.

6 In the Configuration Parameters window, add the following line or, if the file already contains the disable configuration setting, set the value to **TRUE**:

```
isolation.tools.guestlibGetInfo.disable = "TRUE"
```

The default value for the disable setting is **FALSE**. The default setting enables the runtime components. Reinstalling VMware Tools does not affect the disable setting. If you disable the vSphere Guest API and then reinstall VMware tools, the vSphere Guest API continues to be unavailable until you change the `guestLibGetInfo.disable` configuration setting to **FALSE**.

vSphere Guest API Data Types

The vSphere Guest API uses the data types listed in Table 2 to support access to virtual machine data.

Table 2. Data Types

Data Type	Description
`VMGuestLibHandle`	Reference to virtual machine data. `VMGuestLibHandle` is defined in `vmGuestLib.h`.
`VMSessionID`	Unique identifier for a session. The session ID changes after a virtual machine is migrated using VMotion, suspended and resumed, or reverted to a snapshot. Any of these events is likely to render any information retrieved with this API invalid. You can use the session ID to detect those events and react accordingly. For example, you can refresh and reset any state that relies on the validity of previously retrieved information. Use `VMGuestLib_GetSessionId` to obtain a valid session ID. A session ID is opaque. You cannot compare a virtual machine session ID with the session IDs from any other virtual machines.You must always call `VMGuestLib_GetSessionId` after calling `VMGuestLib_UpdateInfo`. `VMSessionID` is defined in `vmSessionId.h`.
`VMGuestLibError`	Status code that indicates success or failure. Each function returns a `VMGuestLibError` code. For information about specific error codes, see "vSphere Guest API Error Codes " on page 13. `VMGuestLibError` is an enumerated type defined in `vmGuestLib.h`.

vSphere Guest API Functions

The vSphere Guest SDK contains the header file `vmGuestLib.h`. This file declares the functions and data types that you use to call the vSphere Guest API. The following sections describe the vSphere Guest API functions:

Context Functions

The vSphere Guest API provides a set of functions that initialize and manipulate the context in which the Guest API operates. Before your application can use the accessor functions to retrieve information about a virtual machine, use the following functions to initialize the vSphere Guest API environment.

1 Call the `VMGuestLib_OpenHandle` function to obtain a handle for accessing information about the virtual machine. The guest library handle is a parameter to every Guest API function.

2 Call the `VMGuestLib_UpdateInfo` function to update the information available through the handle.

3 Call the `VMGuestLib_GetSessionId` function to retrieve a session ID.

Example 1 shows a C code fragment that illustrates the function calls for initialization. (The code fragments in this section do not perform error handling. For information about error handling, see "vSphere Guest API Error Codes " on page 13.)

Example 1. Initializing the vSphere Guest API Environment

```
VMGuestLibHandle glHandle;
VMGuestLibError glError;
VMSessionId sid = 0;
glError = VMGuestLib_OpenHandle(&glHandle);
glError = VMGuestLib_UpdateInfo(glHandle);
glError = VMGuestLib_GetSessionId(glHandle, &sid);
```

You can use the session ID to detect changes that invalidate previously retrieved data. Example 2 shows a code fragment that illustrates how to use the session ID to detect stale data. (The `ResetStats` function in the following fragment represents application code to handle the session change.)

Example 2. Detecting Stale Data

```
VMGuestLibHandle glHandle;
VMGuestLibError glError;
VMSessionId oldSid;
VMSessionId newSid;

/* [...code here would access data based on an existing, valid session ID (oldSid)...] */

/* Update the library, get the session ID, and compare it to the previous session ID */
glError = VMGuestLib_UpdateInfo(glHandle);
glError = GuestLib_GetSessionId(glHandle, &newSid);
if (oldSid != newSid) {
    ResetStats();
    oldSid = newSid;
}
```

Table 3 lists the context functions for creating and releasing handles, updating information, and obtaining session IDs.

Table 3. Open, Close, and Update Functions

Function	Description
`VMGuestLib_OpenHandle`	Gets a handle for use with other vSphere Guest API functions. The guest library handle provides a context for accessing information about the virtual machine. Virtual machine statistics and state data are associated with a particular guest library handle, so using one handle does not affect the data associated with another handle.
`VMGuestLib_CloseHandle`	Releases a handle acquired with `VMGuestLib_OpenHandle`.
`VMGuestLib_UpdateInfo`	Updates information about the virtual machine. This information is associated with the `VMGuestLibHandle`. `VMGuestLib_UpdateInfo` requires similar CPU resources to a system call and therefore can affect performance. If you are concerned about performance, minimize the number of calls to `VMGuestLib_UpdateInfo`. If your program uses multiple threads, each thread must use a different handle. Otherwise, you must implement a locking scheme around update calls. The vSphere Guest API does not implement internal locking around access with a handle.
`VMGuestLib_GetSessionId`	Retrieves the `VMSessionID` for the current session. Call this function after calling `VMGuestLib_UpdateInfo`. If `VMGuestLib_UpdateInfo` has never been called, `VMGuestLib_GetSessionId` returns `VMGUESTLIB_ERROR_NO_INFO`.

Accessor Functions (Virtual Machine)

Accessor functions retrieve information about a virtual machine. When you call an accessor function, you pass a guest library handle (type `VMGuestLibHandle`) to the function. If the function is successful, it returns the requested data as an output parameter. The function return value is an error code (type `VMGuestLibError`) that indicates success or failure. Example 3 shows a C code fragment that illustrates an example of calling `VMGuestLib_GetCpuLimitMHz` to retrieve the processor limit available to the virtual machine.

Example 3. Using an Accessor Function

```
uint32 cpuLimitMHz = 0;
glError = VMGuestLib_GetCpuLimitMHz(glHandle, &cpuLimitMHz);
```

When a call completes successfully, the function returns the value `VMGUESTLIB_ERROR_SUCCESS`. Unsuccessful calls return error codes that contain an appropriate description as part of the error code name. For details, see "vSphere Guest API Error Codes " on page 13.

Call `VMGuestLib_UpdateInfo` once to refresh all statistics before calling an accessor function or a series of accessor functions.

Table 4. Accessor Functions for Virtual Machine Data

Function	Description
`VMGuestLib_GetCpuLimitMHz`	Retrieves the upper limit of processor use in MHz available to the virtual machine. For information about setting the CPU limit, see "Limits and Reservations" on page 12.
`VMGuestLib_GetCpuReservationMHz`	Retrieves the minimum processing power in MHz reserved for the virtual machine. For information about setting a CPU reservation, see "Limits and Reservations" on page 12.
`VMGuestLib_GetCpuShares`	Retrieves the number of CPU shares allocated to the virtual machine. For information about how an ESX server uses CPU shares to manage virtual machine priority, see the *vSphere Resource Management Guide*.
`VMGuestLib_GetCpuStolenMs`	Retrieves the number of milliseconds that the virtual machine was in a ready state (able to transition to a run state), but was not scheduled to run.
`VMGuestLib_GetCpuUsedMs`	Retrieves the number of milliseconds during which the virtual machine has used the CPU. This value includes the time used by the guest operating system and the time used by virtualization code for tasks for this virtual machine. You can combine this value with the elapsed time (`VMGuestLib_GetElapsedMs`) to estimate the effective virtual machine CPU speed. This value is a subset of elapsedMs.
`VMGuestLib_GetElapsedMs`	Retrieves the number of milliseconds that have passed in the virtual machine since it last started running on the server. The count of elapsed time restarts each time the virtual machine is powered on, resumed, or migrated using VMotion. This value counts milliseconds, regardless of whether the virtual machine is using processing power during that time. You can combine this value with the CPU time used by the virtual machine (`VMGuestLib_GetCpuUsedMs`) to estimate the effective virtual machine CPU speed. cpuUsedMS is a subset of this value.
`VMGuestLib_GetHostProcessorSpeed`	Retrieves the speed of the ESX system's physical CPU in MHz.
`VMGuestLib_GetMemActiveMB`	Retrieves the amount of memory the virtual machine is actively using—its estimated working set size.
`VMGuestLib_GetMemBalloonedMB`	Retrieves the amount of memory that has been reclaimed from this virtual machine by the vSphere memory balloon driver (also referred to as the "vmmemctl" driver).
`VMGuestLib_GetMemLimitMB`	Retrieves the upper limit of memory that is available to the virtual machine. For information about setting a memory limit, see "Limits and Reservations" on page 12.
`VMGuestLib_GetMemMappedMB`	Retrieves the amount of memory that is allocated to the virtual machine. Memory that is ballooned, swapped, or has never been accessed is excluded.

Table 4. Accessor Functions for Virtual Machine Data (Continued)

`VMGuestLib_GetMemOverheadMB`	Retrieves the amount of "overhead" memory associated with this virtual machine that is currently consumed on the host system. Overhead memory is additional memory that is reserved for data structures required by the virtualization layer.
`VMGuestLib_GetMemReservationMB`	Retrieves the minimum amount of memory that is reserved for the virtual machine. For information about setting a memory reservation, see "Limits and Reservations" on page 12.
`VMGuestLib_GetMemSharedMB`	Retrieves the amount of physical memory associated with this virtual machine that is copy-on-write (COW) shared on the host.
`VMGuestLib_GetMemSharedSavedMB`	Retrieves the estimated amount of physical memory on the host saved from copy-on-write (COW) shared guest physical memory.
`VMGuestLib_GetMemShares`	Retrieves the number of memory shares allocated to the virtual machine. For information about how an ESX server uses memory shares to manage virtual machine priority, see the *vSphere Resource Management Guide*.
`VMGuestLib_GetMemSwappedMB`	Retrieves the amount of memory that has been reclaimed from this virtual machine by transparently swapping guest memory to disk.
`VMGuestLib_GetMemTargetSizeMB`	Retrieves the size of the target memory allocation for this virtual machine.
`VMGuestLib_GetMemUsedMB`	Retrieves the estimated amount of physical host memory currently consumed for this virtual machine's physical memory.

`VMGuestLib_GetResourcePoolPath`	Retrieves the path name of the resource pool to which the virtual machine belongs on the ESX system where it is running.
	VMGuestLib_GetResourcePoolPath uses an additional parameter to determine the size of the path name output string buffer.
	
	`VMGuestLibError VmGuestLib_GetResourcePoolPath(`
	`VMGuestLibHandle handle,`
	`size_t *bufferSize,`
	`char *pathBuffer );`
	
	`handle` is an input parameter specifying the guest library handle.
	`bufferSize` is an input/output parameter. It is a pointer to the size of the pathBuffer in bytes. If bufferSize is not large enough to accomodate the path (including a NULL terminator), the function returns VMGUESTLIB_ERROR_BUFFER_TOO_SMALL. In this case, the function uses the bufferSize parameter to return the number of bytes required for the string.
	
	`pathBuffer` is an output parameter. It is a pointer to a buffer that receives the resource pool path string. The path string is NULL-terminated.
	For information about using resource pools, see the *vSphere Resource Management Guide*.

For more information about ESX resource management, see the *vSphere Resource Management Guide*, available on the VMware Web site.

Limits and Reservations

You use the Guest API to retrieve information about limits and reservations for CPU and memory. To set limits and reservations, you can use either the vSphere Client or the vSphere API. Setting limits and reservations on a virtual machine ensures that resources are available to that machine and to other virtual machines that draw resources from the same resource pool.

To use the vSphere Client to set limits or reservations:

1 In the vSphere Client window, click on the **Resource Allocation** tab.

2 In either the CPU or Memory section, click **Edit**.

3 In the Virtual Machine Properties Window, click on the **Resources** tab.

4 Select either the CPU or Memory setting.

5 Use the slider controls to set limits or reservations.

For more information, see the help for the vSphere Client.

To use the vSphere API to set limits or reservations, call the **ReconfigVM_Task** method. In the method call, use the **VirtualMachineConfigSpec** data object to set the **cpuAllocation** or **memoryAllocation** property. These properties are of type **ResourceAllocationInfo** type, which has **limit** and **reservation** properties. For more information, see the VMware vSphere API Reference Documentation.

vSphere Guest API Error Codes

Each vSphere Guest API function returns an error code. If successful, the returned error code is `VMGUESTLIB_ERROR_SUCCESS`. If the function is unable to complete its task, the error code provides information for diagnosing the problem.

Use the `VMGuestLib_GetErrorText` function to retrieve the error text associated with a particular error code. When you call the function, pass the error code to the function; `VMGuestLib_GetErrorText` returns a pointer to a string containing the error text.

Example 4 shows error handling. The C code fragment declares a guest library handle and calls the function `VMGuestLib_OpenHandle`. If the call is not successful, the code calls `VMGuestLib_GetErrorText` and displays the error text.

Example 4. Error Handling

```
VMGuestLibHandle glHandle;
glError = VMGuestLib_OpenHandle(&glHandle);
if (glError != VMGUESTLIB_ERROR_SUCCESS) {
   printf("OpenHandle failed: %s\n", VMGuestLib_GetErrorText(glError));
}
```

Table 5 lists all error codes defined for the vSphere Guest API.

Table 5. Error Codes

Error Code	Description
`VMGUESTLIB_ERROR_SUCCESS`	The function has completed successfully.
`VMGUESTLIB_ERROR_OTHER`	An error has occurred. No additional information about the type of error is available.
`VMGUESTLIB_ERROR_NOT_RUNNING_IN_VM`	The program making this call is not running on a VMware virtual machine.
`VMGUESTLIB_ERROR_NOT_ENABLED`	The vSphere Guest API is not enabled on this host, so these functions cannot be used. For information about how to enable the library, see "Context Functions " on page 9.
`VMGUESTLIB_ERROR_NOT_AVAILABLE`	The information requested is not available on this host.
`VMGUESTLIB_ERROR_NO_INFO`	The handle data structure does not contain any information. You must call `VMGuestLib_UpdateInfo` to update the data structure.
`VMGUESTLIB_ERROR_MEMORY`	There is not enough memory available to complete the call.
`VMGUESTLIB_ERROR_BUFFER_TOO_SMALL`	The buffer is too small to accommodate the function call. For example, when you call `VMGuestLib_GetResourcePoolPath`, if the path buffer is too small for the resulting resource pool path, the function returns this error. To resolve this error, allocate a larger buffer.
`VMGUESTLIB_ERROR_INVALID_HANDLE`	The handle that you used is invalid. Make sure that you have the correct handle and that it is open. It might be necessary to create a new handle using `VMGuestLib_OpenHandle`.
`VMGUESTLIB_ERROR_INVALID_ARG`	One or more of the arguments passed to the function were invalid.
`VMGUESTLIB_ERROR_UNSUPPORTED_VERSION`	The host does not support the requested statistic.

Index