Windows

Requirements for Implementing the Microsoft Hypervisor Interface

June 13, 2012

Abstract

This paper provides information about the minimum set of functionality required to support the Microsoft hypervisor interface. It provides details on the mandatory features in the Microsoft-compatible hypervisor interface required for virtualizing Microsoft Windows operating systems, and offers guidelines for virtualization solution developers to determine which optional features may be supported, and the effect of these features on Windows operating systems when running virtualized.

It assumes that the reader is familiar with the Microsoft Hypervisor Top Level Functional Specification.

This information applies to the following operating systems:

Windows 8
Windows Server 2012

References and resources discussed here are listed at the end of this paper.

The current version of this paper is maintained on the Web at:

Requirements for Implementing the Microsoft Hypervisor Interface

Microsoft

Document History

Date	Change
June 13, 2012	First publication

Contents

Introduction

Microsoft publishes the Hypervisor Top-Level Functional Specification (TLFS) for the Microsoft hypervisor, a component of Microsoft Windows Server virtualization since Windows Server 2008. The TLFS specifies the externally visible behavior of the hypervisor. The TLFS can be used to understand the functions of the hypervisor, and enables virtualization solution providers to implement a Microsoft-compatible solution by conforming to the published Microsoft hypervisor interface.

The Microsoft hypervisor interface was designed to allow virtualization providers to implement a minimal subset of the functionality described in the TLFS, and to selectively enable specific features. This paper specifies the minimum set of requirements needed for conforming hypervisors to support virtualizing Windows operating systems, and offers details on the behavior of Windows operating systems in the presence of specific hypervisor provided features beyond the required minimal set that a conforming hypervisor may wish to support.

Supported Windows Operating Systems

The following versions of Windows operating systems support the Hv#1 interface. Note that not all features of the Hv#1 interface may be supported by all Windows versions.

- Windows Vista
- Windows Server 2008
- Windows 7
- Windows Server 2008 R2
- Windows 8
- Windows Server 2012

Requirements for a Minimal HV#1Interface

The minimal interface set required by compliant hypervisors in order to support Windows operating systems when running in a guest virtual machine is summarized below. Details of each requirement are provided in subsequent sections.

- Hypervisor discovery via the CPUID instruction
- Hypervisor CPUID leaves 0x40000000- 0x40000005
- Hypervisor interface signature equal to “Hv#1”
- Partition privileges AccessVpIndex, AccessHypercallMsrs
- Hypervisor synthetic MSRs HV_X64_MSR_GUEST_OS_ID, HV_X64_MSR_HYPERCALL and HV_X64_MSR_VP_INDEX.
- A minimal implementation of the hypercall interface
-

Hypervisor and Microsoft Hypervisor Interface Discovery

During kernel initialization, the Microsoft operating systems listed in this paper perform the following checks to determine if they are running virtualized, the

hypervisor interface present, and which hypervisor features and capabilities may be used.

Detecting the Presence of a Hypervisor

Software determines the presence of a hypervisor through the CPUID instruction. Processors conforming to the Intel® 64 architecture have reserved a feature flag in CPUID Function 0x01 - Feature Information for this purpose. Bit 31 returned in ECX is defined as Not Used, and will always return 0 from the physical CPU. A hypervisor conformant with the Microsoft hypervisor interface will set CPUID.1:ECX [bit 31] = 1 to indicate its presence to software.

Determining Hypervisor Capabilities

When the hypervisor present bit is set, additional CPUID leaves are provided by the hypervisor which will return more information about the hypervisor and its capabilities.

The Intel® 64 architecture reserves CPUID leaves 0x40000000-0x400000FF for use by system software. A Microsoft-compliant hypervisor guarantees leaves 0x40000000 and 0x40000001 are always available

Microsoft Hypervisor Interface Identification

The standard hypervisor CPUID leaf is provided at 0x40000000. When queried, this leaf will return the maximum hypervisor CPUID leaf number, and the vendor ID signature.

Hypervisor Vendor-Neutral Interface Identification

The hypervisor interface identification is provided at CPUID leaf 0x40000001. Hypervisors conforming to the Microsoft hypervisor interface will return the hypervisor interface identification signature ‘Hv#1’ (0x31237648) in CPUID.40000001:EAX.

Hypervisor CPUID Leaves

Refer to the TLFS for the following discussion.

Required Hypervisor CPUID Leaves

The following hypervisor CPUID leaves must be supported by conformant hypervisors. Note that it is generally recommended to return 0 for all CPUID leaves except those that are marked as required in this document, and for those specific features that a conforming hypervisor chooses to implement.

Leaf 0x40000000 — Hypervisor CPUID leaf range and vendor ID signature

Register	Information	Required	Notes
EAX	Hypervisor CPUID leaf range	Yes	
EBX	Vendor ID signature	Yes	Used only for reporting and diagnostic purposes
ECX	Vendor ID signature	Yes	Used only for reporting and diagnostic purposes

Register	Information	Required	Notes
EDX	Vendor ID signature	Yes	Used only for reporting and diagnostic purposes

Notes

This leaf is recommended, and may be used for diagnostic and reporting purposes. For details on reporting hypervisor version information, refer to the TLFS Section 3.5.

Leaf 0x40000002 — Hypervisor system identity

Register	Information	Required	Notes
EAX	Build number	No	
EBX	Bits 31-16: Major Version Bits 15-0: Minor Version	No	Used only for reporting and diagnostic purposes
ECX	Service Pack	No	Used only for reporting and diagnostic purposes
EDX	Bits 31-24: Service Branch Bits 23-0: Service Number	No	Used only for reporting and diagnostic purposes

Notes

This leaf is recommended, and may be used for diagnostic and reporting purposes. For details on reporting hypervisor version information, refer to the TLFS Section 3.5.

Leaf 0x40000003 — Hypervisor feature identification

Register	Information	Required	Notes
EAX	Features available to the partition based upon the current partition privileges	Yes	See details below
EBX	Flags specified at partition creation	Yes	See details below
ECX	Power management related information	No	May be zero
EDX	Miscellaneous features available to the partition	No	May be zero

Notes

CPUID.40000003:EAX and EBX indicate partition privileges and access to virtual MSRs. Conforming hypervisors must implement EAX and EBX as defined below. A conforming hypervisor returning any non-zero value in 0x40000003.EAX or EBX must support the corresponding functionality as defined in the TLFS.

CPUID.40000003:EAX — Partition Privileges

Bit	Description	Required
Bit 0	AccessVpRunTimeMsr	Optional
Bit 1	AccessPartitionReferenceCounter	Optional
Bit 2	AccessSynicMsrs	Optional
Bit 3	AccessSyntheticTimerMsrs	Optional
Bit 4	AccessApicMsrs	Optional
Bit 5	AccessHypercallMsrs	Must be set
Bit 6	AccessVpIndex	Must be set
Bit 7	AccessResetMsr	Optional
Bit 8	AccessStatsMsr	Optional

Bit	Description	Required
Bit 9	AccessPartitionReferenceTsc	Optional
Bit 10	AccessGuestIdleMsr	Optional
Bit 11	AccessFrequencyMsrs	Optional
Bits 12-31	Reserved	

Notes

Partition privileges must be identical for all virtual processors in a partition, and must remain constant for the lifetime of the virtual machine[1].

CPUID.40000003:EBX Feature Identification —Partition Flags

Bit	Description	Required
Bit 0:	CreatePartitions	Must be clear
Bit 1:	AccessPartitionId	Must be clear
Bit 2:	AccessMemoryPool	Must be clear
Bit 3:	AdjustMessageBuffers	Must be clear
Bit 4:	PostMessages	Optional
Bit 5:	SignalEvents	Optional
Bit 6:	CreatePort	Must be clear
Bit 7:	ConnectPort	Optional
Bit 8:	AccessStats	Must be clear
Bit 9-10:	Reserved2	
Bit 11:	Debugging	Optional
Bit 12:	CpuManagement	Must be clear
Bit 13:	ConfigureProfiler	Must be clear
Bit 14-31:	Reserved3	

Notes

These are enlightenment bits which indicate to the guest OS kernel which hypercalls are recommended, in addition to other information. A conforming hypervisor returning any non-zero value in 0x40000004.EAX must support the corresponding functionality as defined in the TLFS.

Leaf 0x40000004 — Enlightenment implementation recommendations

Register	Information	Required	Notes
EAX	Implementation recommendations	No	May be zero
EBX	Recommended number of attempts to retry a spinlock failure	No	Set to 0x0 to disable 0xFFFFFFFF indicates never to retry
ECX	Reserved	No	—
EDX	Reserved	No	—

[1] The AccessPartitionReferenceTsc is exempt from this requirement; see details below in this document.

Notes

These are enlightenment bits which indicate to the guest OS kernel which hypercalls are recommended, in addition to other information. A conforming hypervisor returning any non-zero value in 0x40000004.EAX must support the corresponding functionality as defined in the TLFS.

Leaf 0x40000005 — Implementation limits

Register	Information	Required	Notes
EAX	The maximum number of virtual processors supported	No	May be zero
EBX	The maximum number of logical processors supported	No	May be zero
ECX	Reserved	No	—
EDX	Reserved	No	—

Notes

On Windows operating systems versions through Windows Server 2008 R2, Leaf 0x40000005 — Implementation limits is used for reporting purposes only.

Maximum Supported Virtual Processors

On Windows operating systems versions through Windows Server 2008 R2, reporting the HV#1 hypervisor interface limits the Windows virtual machine to a maximum of 64 VPs, regardless of what is reported via CPUID.40000005.EAX.

Starting with Windows Server 2012 and Windows 8, if CPUID.40000005.EAX contains a value of -1, Windows assumes that the hypervisor imposes no specific limit to the number of VPs. In this case, Windows Server 2012 guest VMs may use more than 64 VPs, up to the maximum supported number of processors applicable to the specific Windows version being used.

However, it is important to note that if more than 64VPs are used, the following hypercalls will not function correctly.

- HvFlushVirtualAddressSpace
- HvFlushVirtualAddressList

Therefore, a conforming hypervisor reporting -1 in CPUID.40000005.EAX must not recommend these hypercalls (i.e., CPUID.40000004.EAX:1-2 must be cleared).

Leaf 0x40000006 — Implementation hardware features

Register	Information	Required	Notes
EAX	Implementation recommendations	No	May be zero
EBX	Reserved	No	—
ECX	Reserved	No	—
EDX	Reserved	No	—

Hypervisor Synthetic MSRs

The Microsoft hypervisor interface defines a number of synthetic MSRs that are available to guest software, depending on the partition privileges. Windows operating systems supporting the Hv#1 interface require the following synthetic MSRs to be present in conforming hypervisors.

Hypervisor Synthetic MSRs

MSR Number	MSR Name	Required
0x40000000	HV_X64_MSR_GUEST_OS_ID	Yes
0x40000001	HV_X64_MSR_HYPERCALL	Yes
0x40000002	HV_X64_MSR_VP_INDEX	Yes

Virtual Processor Index

Microsoft Windows operating systems running in a virtual machine identify virtual processors using a VP index retrieved from synthetic MSR 0x40000002. A conforming hypervisor must supply VP indices, and all VP indices must be unique.

Hypercall Interface

A conforming hypervisor must support mapping a hypercall page within the guest's GPA space. The hypercall page must be both readable and executable, and the contents of the mapped hypercall code page must not change without an un-map/map transition. The hypercall page does not actually have to cause a hypervisor transition. Note that Windows Kernel Patch Protection (aka Windows PatchGuard) protects the contents of the hypercall code page.

All enlightened versions of Windows operating systems invoke guest hypercalls on the basis of the recommendations presented by the hypervisor in CPUID.40000004:EAX. A conforming hypervisor must return HV_STATUS_NOT_IMPLEMENTED for any unimplemented hypercalls. If a hypervisor does not wish to handle any hypercalls, it may implement the following hypercall code page minimal sequence.

```
mov eax, 0x02 ; HV_STATUS_INVALID_HYPERCALL_CODE
mov edx, 0
ret
```

Optional Partition Privileges

Conforming hypervisors may elect to implement select features beyond the minimal set of requirements described in this document. Examples of such features are:

- The partition reference TSC enlightenment
- Enabling relaxed timing in the guest OS
- The virtual guest idle state

Each of these is discussed in greater detail below.

Partition Reference Time Enlightenment

The partition reference time enlightenment is documented in the TLFS section 15.4. A conforming hypervisor may also implement similar support, as long as the implementation provides the expected semantics. A conforming hypervisor must provide the HV_X64_MSR_REFERENCE_TSC and HV_X64_MSR_TIME_REF_COUNT MSRs.

The partition reference time enlightenment is supported on the following Windows versions:

- Windows 7
- Windows 7 SP1
- Windows Server 2008 R2
- Windows Server 2008 R2 SP1

In order to use the partition reference time enlightenment, the Windows guest OS partition must hold the following partition privileges:

- AccessPartitionReferenceCounter privilege (CPUID.40000003.EAX:1=1). A hypervisor that provides this privilege must provide the HV_X64_MSR_TIME_REF_COUNT MSR.
- On systems with a constant rate TSC (C-state invariant TSC, or iTSC, the AccessPartitionReferenceTsc privilege (CPUID.40000003.EAX:9=1). The hypervisor must provide the HV_X64_MSR_REFERENCE_TSC MSR and allow mapping the reference TSC page.

If the Reference TSC and Reference Time enlightenments are present, Windows requires that:

- All TSCs must be sync'd across all processors
- If support for iTSC is advertised (CPUID.80000007.EDX:8=1), the hypervisor must ensure the TSC rate remains constant for the lifetime of the VP, across all partition state change operations such as partition Saves, Restore, migration of the partition to a different virtualization host, etc. If the iTSC is present, Windows will use RDTSC directly as the system time source backing the QueryPerformanceCounter function call2.

Reference Time Enlightenment and Virtual Machine Migration

If a Windows VM which supports the Reference Time Enlightenment starts on an invariant TSC system and then is moves to a system without an invariant TSC, it will use the fallback mechanism described in the TLFS v2.0 Section 15.4.3.3 *Reference TSC during Save/Restore and Migration*, wherein the VM will revert to using the virtual ACPI Power Management Timer (PM Timer). However, if the VM starts on a non-invariant TSC system and moves to an invariant TSC system, it will not re-enlighten itself to detect the presence of the partition reference time enlightenment. In all

[2] Applies to Windows Server 2008 R2 and later.

cases, the underlying hypervisor must preserve the reference time across all virtual machine migrations or state changes.

Use Relaxed Timing

Hypervisor CPUID leaf CPUID.0x40000004.EAX:5 supplies a recommendation that the guest OS should use relaxed timing. When Windows operating systems supporting the Hv#1 interface detect that this bit is set[3], they disable both clock interrupt and DPC watchdog timeouts. This helps avoid false positive triggers of these watchdog timers due to delays in delivering interrupts or scheduling virtual processors that might be introduced on a heavily loaded or over-subscribed virtualization platform.

Starting with Windows Server 2012 and Windows 8, Windows will use relaxed timing if any hypervisor is present (i.e., if CPUID.1:ECX [bit 31]=1. If the hypervisor declares support for the Hv#1 interface, then Windows' use of relaxed timing will follow the recommendation in CPUID.0x40000004.EAX:5.

Virtual Guest Idle State

Windows 7 and Windows Server 2008 R2 introduced support for several processor power management enhancements, including Intelligent Timer Tick Distribution (ITTD)[4]. ITTD helps to extend the amount of time that processor cores remain in the idle state by not interrupting all cores in the system when the periodic timer interrupt is delivered. Only the base service processor (BSP) receives every timer tick interrupt, which it optionally delivers to secondary processor cores. On virtualized systems, ITTD helps realize reduced interrupt traffic and longer idle periods.

Windows cannot use ITTD when entering the ACPI C1 processor idle state due to the entry semantics of the C1 state. However, Windows operating systems do not support legacy ACPI processor idle sleep states greater than the ACPI C1 state in the presence of the Hv#1 hypervisor interface.

To enable the use of processor power management enhancements such as ITTD, the TLFS v2.0 defines a Virtual Processor Idle Sleep State in Section 10.1.4. Supporting the virtual guest idle state requires the AccessGuestIdleMsr privilege (CPUID.40000003:EAX:10=1), and support for the HV_X64_MSR_GUEST_IDLE MSR.

Miscellaneous Implementation Notes

This section discusses general notes on implementation details when Windows operating systems are run in virtual machines.

Inter-Processor Interrupts (IPI)

Conforming hypervisors must provide special semantics for self-IPIs. Following any guest instruction which has the effect of sending an IPI (e.g., a write to the virtual APIC's Interrupt Command Register, or a write to the HV_X64_MSR_ICR MSR[5]), if the sending VP is included in the destination of the IPI, the sending VP must receive the

[3] Not supported on Windows Vista RTM

[4] Refer to "*Processor Power Management in Windows 7 and Windows Server 2008 R2*" in the References section in this document.

[5] If implemented by the hypervisor

interrupt before the next guest instruction is executed. This ensures that if the sending VP is ready to service the interrupt it will be serviced immediately, before any other guest instructions are executed.

Microsoft Hyper-V Virtual Machine Bus

Third party virtualization solutions must not claim support for the Microsoft Hyper-V Virtual Machine Bus (VMBus) device in the virtual BIOS ACPI namespace. The VMBus device should be correctly disabled on any V2V or P2V conversion.

Resources

Hypervisor Functional Specification v2.0a: For Windows Server 2008 R2
http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=18673

Windows ACPI Emulated Devices Table
http://msdn.microsoft.com/en-us/windows/hardware/gg487524

Processor Power Management in Windows 7 and Windows Server 2008 R2
http://www.microsoft.com/whdc/system/pnppwr/powermgmt/ProcPowerMgmtWin7.mspx