VMware VMotion and CPU Compatibility

VMware® Infrastructure 3

VMware VMotion technology allows users to migrate running virtual machines between compatible physical servers with zero downtime. To ensure successful migration and subsequent functioning of the virtual machine, you must respect certain compatibility constraints. This information guide describes CPU compatibility issues and outlines some CPU compatibility checks that VMware VirtualCenter performs before allowing migration with VMotion. It describes why some CPU compatibility constraints make VMotion possible only between certain revisions of CPUs. The appendices detail some differences in features and extensions in current CPUs and describe procedures you can use to relax some CPU compatibility constraints to facilitate VMotion.

This guide is primarily intended for:

- Purchasing managers and system administrators who configure and purchase new servers for a data center upgrade
- System administrators who want to perform migrations with VMotion or want to relax CPU compatibility constraints in order to circumvent VMotion checks
- High level IT staff who make decisions on potential data center upgrades
- Anyone who wants to understand how CPU compatibility constraints can affect live migration of virtual machines

This guide covers the following topics:

Introduction to VMware VMotion

The VMware VMotion feature, part of VirtualCenter 1.0 and later releases, allows you to migrate running virtual machines from one physical machine to another with no perceivable impact to the end user. You can use VMotion to upgrade and repair servers without any downtime or disruptions and also to optimize resource pools dynamically, resulting in an improvement in the overall efficiency of a data center.

Key Concepts

The following terms and concepts are important to understanding the VMotion technology and the compatibility constraints that affect use of VMotion.

Host A physical server that is part of the VMware Infrastructure hardware resource pool. This host may be used to run one or more virtual machines.

Migration of a virtual machine The process of moving an entire virtual machine from one host to another. Complete virtualization of all components of a machine, such as CPU, BIOS, storage disks, networking, and memory allows the entire state of a virtual machine to be captured by a set of data files. Therefore, moving a virtual machine from one host to another is, in essence, a specialized data transfer between two hosts. Based on the state of the virtual machine during migration, the migration is referred to as cold or hot.

- **Cold migration:** Migration of a virtual machine that has been powered off on the source host. The virtual machine is powered on again on the destination host after the transfer of the virtual machine state is completed.
- **Hot migration:** Migration of a virtual machine that is powered on. The virtual machine (and applications) previously running on the source host continue execution on the destination host, without detecting any changes, after the hot migration is complete. Based on availability of the virtual machine during migration, hot migration falls into the following subcategories:
  - **Suspend/resume migration:** Involves suspending a running virtual machine, then resuming the suspended virtual machine on a different host.
  - **Live migration (VMotion):** Migration of a running virtual machine from a source host to a destination host without any disruptions (downtime) to the running virtual machine.

CPU microarchitecture The internal architecture, including design and interaction of different components, of the CPU. The microarchitecture of a CPU is an implementation of a particular instruction set architecture (ISA). Different microarchitectures might implement the same ISA irrespective of CPU vendor. For example, Intel Core-based CPUs differ from their P4 counterpart in microarchitecture, but implement the same x86 ISA.

Privileged code Set of instructions that run at the highest privilege level and can therefore perform operations critical to the functioning of a machine and may affect all applications. On x86 CPUs, the highest privilege level is 0. Typically, operating system related (kernel) instructions run at the highest privilege level. Any instruction may run in the privileged mode if the operating system specifies that it do so.

Nonprivileged code Instructions that belong to application software. Not all instructions can run in nonprivileged mode. Applications sometimes need to perform privileged operations and use services (such as APIs) provided by the operating system to achieve this. On x86 systems, applications typically run at a privilege level of 3, though it is correct for applications running at privilege levels of 1 and 2 to be categorized as nonprivileged code.

CPUID instruction An x86 assembly instruction used for processor and feature identification. This instruction is the prescribed and standardized method for software to determine the set of features and instructions available on the current CPU.

Applications of VMotion

The ability to migrate a running virtual machine across different hosts without any perceivable impact to the end user opens up a wide array of applications. Some important applications of VMotion are:

- **Hardware maintenance:** VMotion allows you to repair or upgrade the underlying hardware without scheduling any downtime or disrupting business operations.
- **Optimizing hardware resources:** VMotion lets you move virtual machines away from failing or underperforming hosts.
- **VMware product upgrades:** VMotion, coupled with disk migration capabilities, allow you to upgrade from an older version of VMware ESX. You can also use this feature to upgrade between incompatible versions of the VMware Virtual Machine File System (VMFS) while keeping the virtual machine live.

- **VMware Distributed Resource Scheduler:** VMware Distributed Resource Scheduler (DRS) continuously monitors utilization across resource pools and allocates resources among virtual machines based on current needs and priorities. When virtual machine resources are constrained, DRS makes additional capacity available by migrating live virtual machines to a less-utilized host using VMotion.

A more detailed list of advantages and applications of VMotion is available in the VirtualCenter VMotion feature introduction on the VMware Web site (see "References" on page 7 for a link).

Requirements for VMotion

In order to facilitate VMotion between hosts, each host must meet certain basic requirements:

- **Datastore compatibility:** The source and destination hosts must use shared storage. You can implement this shared storage using a SAN or iSCSI. The shared storage may use VMFS or shared NAS. Disks of all virtual machines using VMFS must be available to both source and target hosts.
- **Network compatibility:** VMotion itself requires a Gigabit Ethernet network. Additionally, virtual machines on source and destination hosts must have access to the same subnets, implying that network labels for each virtual Ethernet adapter should match. You should configure these networks on each ESX host.
- **CPU compatibility:** The source and destination hosts must have compatible sets of CPUs. This requirement is discussed in detail in this guide.

This guide outlines the CPU compatibility requirements for VMware VMotion. Refer to the VMware Infrastructure 3 *Basic System Administration* guide (see "References" on page 7) for more information about other compatibility requirements for VMotion.

CPU Compatibility Requirements for VMotion

VirtualCenter performs CPU compatibility checks to ensure that a virtual machine can perform normally on the destination host after migration with VMotion. It expects the CPUs on the source and destination hosts to provide the same set of features to the virtual machine so the live virtual machine and accompanying applications do not crash. The same CPU compatibility checks apply to suspend/resume migration. This section examines the reasons that CPU compatibility checks for VMotion are necessary.

Rapid Evolution of CPUs and the x86 Architecture

CPUs frequently undergo tremendous improvements, many of which require augmentations to the x86 architecture itself. For example, the addition of SSE3 instructions in Intel CPUs facilitates faster floating-point operations, which in turn results in better performance for multimedia applications. Several such additions support specific classes of applications. Furthermore, different CPU vendors may choose to extend the x86 instruction set in different ways based on the applications they decide to target. For example, many AMD CPUs support a set of 3DNow! instructions for multimedia processing capabilities. Intel CPUs address the same target applications with a set of SSE instructions. These sets of instructions differ in many respects.

To summarize:

- Manufacturers constantly revise CPUs.
- Newer revisions of CPUs might have new instructions or features that were unavailable in older revisions.
- Different CPU vendors may choose to implement different instructions to address the same concerns.

Feature Determination and CPUID

Software can use the `CPUID` assembly language instruction to obtain the exact set of features supported by an x86 CPU. Details about the CPUID instruction are available in "Appendix A: CPUID and x86 Implementation Differences" on page 7. If you want to determine the set of features supported by a CPU, you can use the bootable CPUID image available from the VMware Web site (http://www.vmware.com/download/shared_utilities.html). For information on using this CPUID ISO image, see "Appendix C: Relaxing VMware VMotion CPU Compatibility Constraints" on page 11. You can also use the Managed Object Browser feature of ESX, which you can access using a Web browser, to determine the set of features available on the host CPU.

CPU vendors recommend that applications use the `CPUID` instruction for feature determination. Well-behaved system and application software does so. A typical application determines the set of available CPU features by executing `CPUID` only once as it is starting. Based on the results, the application might continue to use some of these features until it is shut down.

However, not all applications are well behaved. Since the `CPUID` instruction was introduced on x86 CPUs only in the early 1990s, legacy software might not follow this recommended method of feature detection and might use custom methods instead. Applications might use the support for a special instruction on a particular CPU as an indication of the availability of features represented by that instruction. For instance, existence of the `ADDPD` assembly instruction could imply presence of all SSE2 instructions. If the application executes this special instruction optimistically and receives an UnDefined instruction exception (`#UD`) from the CPU, it handles this exception and assumes the class of features is unsupported. Because the application handles and consumes the exception, this process may be totally transparent to the end user. Such try-catch-abort exceptions are built into older application software, and the applications may continue to work normally even if they do not adhere to methods recommended by CPU vendors.

Therefore, it is important to note the following:

- A typical application queries the CPU for supported features only once—during startup or initialization—and continues to use these features (if appropriate) as long as the application is active.
- Though there are recommended methods of determining CPU features (via a `CPUID` instruction), different applications may have different methods of determining CPU support for certain features or instructions.

For information on writing applications that test CPU features in a way that facilitates use of VMotion, see the VMware knowledge base article "Testing and Using New Features in CPUs" (for a link, see "References" on page 7.)

High-Performance Virtualization

High-performance virtualization requires delivering virtualization at near-native performance. For this to be technically feasible while maintaining correctness, it is necessary to execute many instructions directly on the underlying hardware. Utilizing underlying hardware resources to facilitate high performance execution is true of any virtualization layer. This ensures applications running inside a virtual machine do not encounter major slowdowns. The virtualization layer intercepts only certain select instructions that really require interception and subsequent emulation.

The x86 ISA allows only instructions belonging to the operating system or kernel to be intercepted by the traditional virtualization layer—that is, a virtualization layer that does not rely on hardware support for virtualization or on paravirtualization. These instructions run at the highest privilege level, making them privileged code. Typically, instructions that run at lower privilege level (nonprivileged code) are passed through to the underlying hardware unimpeded. The `CPUID` instruction can be executed by the operating system (privileged code) as well as by applications (nonprivileged code). If this instruction is part of an application, it executes directly on the underlying hardware, thereby giving an application a list of features available on the underlying hardware. If `CPUID` is part of privileged code, the virtualization layer can intercept it if emulation is needed. This methodology arises from attempting to virtualize the inherently nonvirtualizable x86 ISA (see "Formal Requirements for Virtualizable Third Generation Architectures" in "References" on page 7).

Therefore, even when applications are running within a virtual machine, the following may be true:

- Applications have many of their instructions running directly on underlying hardware.
- Instructions such as `CPUID` might be executed as a nonprivileged instructions and therefore run directly on the underlying hardware, potentially tying an application to the set of features supported by the CPU on a particular host.

Live Migration

Migrating an entire virtual machine while it is running is useful only if applications continue to operate normally after migration. Applications typically include nonprivileged instructions, which might execute directly on the underlying hardware CPU. If applications are to continue performing normally after a live migration, they need to execute instructions and utilize features that were available on the source host hardware even when they are running on the destination host hardware. This is required even though these applications are executing within the same virtual machine.

Cold migration, on the other hand, ensures that the virtual machine and underlying applications restart after migration. Because the applications restart, they initialize and query the CPU for available features. This implies the applications utilize only instructions and features supported by the destination host hardware (where the virtual machine is powered on).

x86 ISA Implementation Differences

Most additions to the x86 ISA will continue to be part of the architecture to ensure backward compatibility. Addition of features and instructions in newer CPUs and differences in implementation of any particular feature by different CPU vendors affect all applications that use them. Porting applications to utilize features specific to a CPU vendor is an arduous task, and the rapid pace of CPU improvements makes this harder. For example, recent Intel CPUs support the SSSE3 instruction set. If applications are tuned to utilize and benefit from these extensions, they need to be designed to account for the fact that such a feature may not exist on CPUs manufactured by other vendors. It may also be the case that other vendors offer some SSSE3-like extensions that serve the same purpose but could be named and implemented differently.

"Appendix A: CPUID and x86 Implementation Differences" on page 7 lists some features and extensions that illustrate implementation differences between CPU vendors or even within CPU revisions. If a feature or an instruction set extension exists on a source host machine and is made available to virtual machines, VirtualCenter requires these extensions to be available on the destination host for compatibility during VMotion.

VirtualCenter Compatibility Checks for VMotion

Unless you are using enhanced VMotion compatibility, VirtualCenter must match up the features of the source and destination host CPUs to ensure that a virtual machine and its applications will operate normally after migration with VMotion. VirtualCenter uses features reported by the `CPUID` instruction to do so. VirtualCenter matches features between the source and destination host CPUs to ensure that all features are compatible. However, VirtualCenter also has a set of masks you can apply to hide features reported by the `CPUID` instruction, so that the query can view fewer features. These masks are used only when `CPUID` is executed as a part of privileged code. They ensure that only features that matter to the execution of the virtual machine are compared. See "Appendix B: CPU Identification Masks" on page 10 for more information about default and override masks.

Because these features are present in hardware, nonprivileged code querying `CPUID` is not affected by any part of the virtualization layer, thus these masks are not applied when the nonprivileged code makes the query. Therefore, applying `CPUID` override masks as shown in the example in Appendix C circumvents CPU compatibility checks for VMotion, but an application running in a virtual machine still might crash if it does not find the set of instructions that were available to it on the source host hardware. This problem is effectively managed by hardware support for live migration, as described in "Enhanced VMotion Compatibility" on page 6.

The above facts and examples demonstrate why CPU compatibility checks are necessary for migration with VMotion. These checks not only ensure that the migration succeeds, they also ensure the applications running inside a virtual machine continue to operate normally on the destination host hardware. These checks for CPU compatibility before allowing VMotion help make ESX a stable, reliable platform.

Enhanced VMotion Compatibility

Because new features are constantly incorporated into new CPUs, you are likely to encounter incompatibilities and therefore face problems while attempting migration with VMotion. Such CPU incompatibilities limit VMotion to a certain range of CPUs. VMware has investigated the issues and concluded that there is no software-only solution to this problem. To minimize exacerbation of the compatibility problem with time, VMware has worked with CPU vendors to facilitate live migration of virtual machines across different CPU revisions.

VMware Enhanced VMotion Compatibility (EVC)—available in VMware Infrastructure 3 beginning with version 3.5 Update 2—facilitates VMotion between different CPU generations, taking advantage of Intel Flex Migration and AMD-V Extended Migration technologies. When enabled for a cluster, EVC ensures that all CPUs within the cluster are VMotion compatible. CPUs starting with Intel 45nm Core 2 (Penryn) and AMD Second Generation Opteron (revision E or F) incorporate FlexMigration and Extended Migration technologies, respectively.

The EVC feature allows the virtualization layer to mask or hide certain features by modifying the semantics of the CPUID instruction and hides certain `CPUID` feature bits, even from nonprivileged code. For example, with support from hardware, the virtualization layer modifies the semantics of the `CPUID` instruction to mask or hide the SSE4.1 feature from any code (privileged or nonprivileged) to make CPUs differing in this feature compatible for VMotion. Specifics on CPU compatibility with the Enhanced VMotion Compatibility feature are available in the *Basic System Administration* guide for each ESX release starting with version 3.5 Update 2.

EVC utilizes hardware support to modify the semantics of the `CPUID` instruction only. It does not disable the feature itself. For example, if an attempt to disable SSE4.1 is made by applying the appropriate masks to a CPU that has these features, this feature bit indicates SSE4.1 is not available to the guest or the application, but the feature and the SSE4.1 instructions themselves (such as `PTESE` and `PMULLD`) are still available for use. This implies applications that do not use the `CPUID` instruction to determine the list of supported features, but use try-catch undefined instructions (`#UD`) instead, can still detect the existence of this feature.

Therefore, for EVC to be useful, application developers must adhere to recommended guidelines on feature detection. CPU vendors recommend that software programmers query `CPUID` prior to using special instructions and features available on their CPUs. If this guideline is followed by programmers, EVC is a reliable mechanism for live migration of x86 virtual machines across varied hardware. Thus, you can use EVC to enable an entire cluster to use the same set of basic features, allowing migration with VMotion across any two nodes in the cluster. VirtualCenter can also set up new hardware add-ons to the cluster and apply these masks.

Conclusion

Virtualization is still a relatively new technological innovation, and the area of live migration of virtual machines is still in its infancy. VMware VMotion is an extremely useful and critical feature in data centers to ensure business continuity, resource optimization, and a host of other benefits. VirtualCenter performs numerous compatibility checks before allowing migration with VMotion. Some stringent CPU compatibility checks for VMotion are necessary for proper functioning of a virtual machine after migration. Though users can override these checks and complete VMotion by applying appropriate masks, the virtual machines and applications may not function properly if they rely on features particular to the underlying hardware. The x86 CPU vendors did not initially envision live migration of virtual machines and design CPUs to support it. VMware has worked with CPU vendors to support such features, taking advantage of Intel Flex Migration and AMD-V Extended Migration technologies. With Enhanced VMotion Compatibility, VMware Infrastructure works in conjunction with hardware to support live migration of virtual machines in a wider range of environments. Effective use of enhanced VMotion also depends on using application software that follows recommended guidelines to support CPU feature detection.

References

VMware Infrastructure Documentation

- VMware Infrastructure 3 Online Library
  http://www.vmware.com/support/pubs/vi_pubs.html
- VMware Infrastructure 3 *Basic System Administration*
  http://www.vmware.com/pdf/vi3_35/esx_3/r35u2/vi3_35_25_u2_admin_guide.pdf

VMware Knowledge Base Articles

- "Testing and Using New Features in CPUs"
  http://kb.vmware.com/kb/1005763
- "VMotion and CPU Compatibility FAQ"
  http://kb.vmware.com/kb/1005764
- "VMotion CPU Compatibility Requirements for Intel Processors"
  http://kb.vmware.com/kb/1991
- "VMotion CPU Compatibility Requirements for AMD Processors"
  http://kb.vmware.com/kb/1992
- "VMotion CPU Compatibility —Migrations Prevented Due to CPU Mismatch—How to Override Masks"
  http://kb.vmware.com/kb/1993

CPU Vendor Documentation

- AP-485 Intel Processor Identification and the CPUID Instruction
  http://developer.intel.com/design/processor/applnots/241618.htm
- AMD "CPUID Specification"
  http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/25481.pdf
- "AMD Processor Recognition" application note
  http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/20734.pdf

Other References

- Bootable CPUID image
  http://www.vmware.com/download/shared_utilities.html
- "Formal Requirements for Virtualizable Third Generation Architectures," Gerald J. Popek and Robert P. Goldberg, *Communications of the ACM* 17 (7): 412 -421 (1974)
  http://portal.acm.org/citation.cfm?id=361073
- VMware VMotion feature introduction
  http://www.vmware.com/products/vi/vc/vmotion_features.html

Appendix A: CPUID and x86 Implementation Differences

To understand CPU compatibility requirements for VMotion, consider how applications determine the set of features supported on an x86 CPU. The `CPUID` instruction provides applications with a standardized method for processor and feature identification. The availability of a particular feature is indicated by the value of the corresponding feature bit in the general-purpose register. The `CPUID` instruction requires a level number or an extended level number in the EAX register in order to execute. Executing `CPUID` with a level number in EAX returns information in the general-purpose registers (such as EAX, EBX, ECX, and EDX). The regular level numbers yield general information, such as CPU vendor, family, model, and stepping. The extended level numbers yield information on specific extended features supported by that CPU vendor for a family, generation, or revision. The extended levels also utilize general-purpose registers to convey extended feature information. The level number in EAX is sometimes referred to as function number or leaf.

A regular function number in EAX is of the form 0x000000XX, where XX is a hexadecimal value. For example, with 0h (0x00000000) present in EAX, `CPUID` returns the following on an Intel CPU:

Table 1. Sample Returns from CPUID

Value in EAX	Returned Values	Meaning
0x00000000	EAX	Largest regular function number in EAX supported by this CPU
	EBX	Processor vendor string
	ECX	
	EDX	

The extended level number lists specific feature information. An extended level number in EAX is of the form 0x800000XX, where XX is a hexadecimal value. For example, to detect NX bit support in an Intel or an AMD CPU, EAX is loaded with 0x80000001h and the `CPUID` instruction is executed. The value of bit 20 in the EDX register indicates whether the CPU supports NX (0 is unsupported, 1 is supported). This guide uses the following notation to examine the value of a particular `CPUID` bit:

```
CPUID level <feature/extended feature> <Register>:<bit>
```

For instance, for NX support, `CPUID level 0x80000001 EDX:20` should be set.

The family or extended family, model, and stepping bits simply describe a CPU, whereas the feature and extended feature bits provide information for use by various kinds of software. The `CPUID` instruction by itself does not require any special privileges and therefore the operating system (privileged mode) or any other application (nonprivileged mode) can execute it.

Specific features and extended features returned by the `CPUID` instruction are closely tied to the CPUs and are decided by CPU vendors. For information about `CPUD` on Intel CPUs, refer to "AP-485 Intel Processor Identification and the CPUID Instruction," and for information about CPUID on AMD CPUs, refer to the AMD "CPUID Specification" (see "References" on page 7 for links).

x86 Feature Bits and Implementation Differences

This section lists some instructions and features that demonstrate implementation differences between vendors or even within CPU revisions—instructions and features that VirtualCenter checks for compatibility before allowing migration with VMotion. This list is based on problems commonly encountered during CPU compatibility checks when performing VMotion. It is not exhaustive and should be treated as a set of examples. This list is based only on current CPUs. Details may vary and change with newer revisions of CPUs.

- **CPU vendor string:** The CPU vendor string is present in CPUID level 0x0 EBX, ECX, and EDX registers. The default VirtualCenter masks require this string to match between source and destination CPUs for VMotion compatibility. This match is required because different vendors may choose to interpret, implement, and extend the x86 ISA differently.
- **CPU family, extended family:** The CPU family is present in CPUID level 0x1 EAX:8 –11. The extended family field is present in CPUID level 0x1 register EAX:20–27. A CPU family is completely defined by the combination of family (sometimes referred to as the base family) and the extended family. The default VirtualCenter masks require this value to match because different CPU families, even from the same vendor, have different supported instruction sets and features.
- **RDTSCP:** This instruction can be executed only by privileged code and is specific to AMD CPUs. If this instruction is supported by the CPU, it is indicated by CPUID level 0x80000001h EAX:27 being set. VirtualCenter supports masking this feature bit and therefore disabling it for applications running inside VMware virtual machines. This allows successful validation of CPU compatibility checks and facilitates VMotion.
- **Streaming SIMD extensions (SSE) and 3DNow!:** These sets of instructions are extensions to the x86 ISA allowing CPUs to process single instruction multiple data (SIMD) instructions that are widely used in multimedia applications. Different CPU vendors chose to extend the x86 instruction set in different ways. Intel designed the SSE instructions. AMD designed the 3DNow! extensions.

- **SSE3:** SSE3 is the third generation of SSE added to the x86 instruction set. Support for SSE3 is indicated by the value of CPUID level 0x1 ECX:0. If this feature is available on a machine, it could potentially be used by certain multimedia applications. If such applications are to continue operating normally after migration with VMotion, this feature must be available in hardware on the destination host. VirtualCenter therefore requires that this feature match between the source and destination hosts for VMotion compatibility.
- **SSSE3:** SSSE3 is the set of supplemental SSE3 instructions added on CPUs based on the Intel Core architecture. This feature is currently available only on Intel CPUs. It is considered a revision of the SSE3 instruction set. As with the SSE3 instructions, if this feature is available on the source host, it must be available on the destination host. Existence of this feature is confirmed by the value of CPUID level 0x1 ECX:9.
- **SSE4.1:** At the time we wrote this guide, SSE4.1 instructions were available only on Intel CPUs. This feature is the first subset of the fourth revision of the SSE instructions and is available for applications to use, if supported by the CPU. This feature is indicated by the value of CPUID level 0x1 ECX:19.
- **SSE4.2:** At the time we wrote this guide, SSE4.2 was yet to be introduced. It will be the second subset of the fourth generation of SSE instructions. This feature is indicated by the value of CPUID level 0x1 ECX:20.
- **3DNow!:** The 3DNow! instructions process vector floating point operations, which aid multimedia processing. These instructions were first made available by AMD. Presence of this feature is detected by the value of CPUID level 0x80000001 EDX: 31.
- **3DNow! Extensions:** Extensions to the 3DNow! instructions were introduced by AMD. This feature is detected by the value of CPUID level 0x80000001 EDX:30.

- **No eXecute (NX) or eXecute Disable (XD):** This feature allows privileged code (kernel level) to mark certain sections of memory as code and others as data. This marking prevents the CPU from executing data regions of memory, one method used by malicious software to take unauthorized control of a machine. Such usurpation is referred to as a buffer overflow attack. Presence of this feature is indicated by CPUID level 0x80000001 EDX:20. Because this feature may be used only by operating systems (privileged code), VirtualCenter allows users to disable this feature and hide it from the guest operating system even if it is available on hardware.
- **Long mode support:** CPUs supporting long mode allow 64-bit instructions (64-bit mode) as well as 32-bit instructions (compatibility mode) to execute. This support is indicated by CPUID level 0x80000001 EDX:29. If a CPU supports long mode, capable applications execute 64-bit instructions. VMware products require this feature to be present when you create a 64-bit virtual machines capable of running a 64-bit guest operating system. VirtualCenter attempts to match this feature bit when performing compatibility checks before allowing migration with VMotion. The compatibility checks for long mode support are based on the configured operating system and settings for the virtual machine. VMware products support long mode only if hardware virtualization support (VT on Intel CPUs and AMD-V on AMD CPUs) is also available on the host.
- **CMPXCHG instructions:** The 8-byte and 16-byte compare and exchange instructions are new additions available on certain newer CPUs. Their presence is indicated by the value of CPUID level 0x1 EDX:8 (for CMPXCHG8B) and CPUID level 0x1 ECX:13 for CMPXCHG16B. These instructions may be used by any application, if available. Therefore, VirtualCenter must match this feature between source and destination hosts before allowing migration with VMotion.
- **FFXSR:** The FXSAVE and FXRSTOR instructions save and restore x87, MMX, and XMM registers used in floating point operations. Current AMD CPUs also have optimized versions of these instructions: fast FXSAVE and fast FXRSTOR (FFXSR), indicated by CPUID level 0x80000001 EDX:25, which do not save and restore XMM registers in the 64-bit mode when executed by privileged code. This feature bit is unused (reserved) on Intel CPUs. Because applications can query the CPU for these features and use them, VirtualCenter requires these features to match between source and destination when it performs VMotion compatibility checks.

- **Prefetch instructions:** AMD CPUs support PREFETCH and PREFETCHW instructions to load data into the CPU's L1 cache. Applications may use these instructions, if available. These instructions are available only if CPUID level 0x80000001 ECX:8 is set. Intel CPUs do not currently support this instruction, and this bit is reserved.
- **MONITOR/MWAIT:** Intel CPUs introduced the MONITOR and MWAIT instructions along with the SSE3 extensions. They are primarily used for thread synchronization and therefore have generic usage. Availability of these instructions is indicated by CPUID level 0x1 ECX:3. Recent AMD CPUs now also support these instructions.
- **Model-specific registers (MSR):** Processor implementations provide certain control registers that allow applications to use features specific to a processor. Since MSRs are specific to a processor model and implementation, they may not be available on other models of the same processor family or other models from the same vendor.

Appendix B: CPU Identification Masks

VirtualCenter queries the host CPU for the set of available features. It then applies a default mask to this value. The default masks are created based on the guest operating system and determine the set of features available to the guest operating system. Privileged code executing `CPUID` will be able to see only the features that are not masked. In order to give system administrators the flexibility to control the features their virtual machine and the guest operating system observe, VirtualCenter allows you to override its default masks with override masks. You can mask out any feature bit at any `CPUID` level. VirtualCenter combines its default mask with your override mask and applies the combined mask to the value returned by `CPUID` to display the final set of features available to the virtual machine. Note that in the absence of support for enhanced VMotion, nonprivileged code executing `CPUID` runs directly on hardware, overriding any masks defined in VirtualCenter. Table 2 shows the notations used in the override and default masks.

Table 2. Notations Used in CDP Feature Masks

Mask	Explanation
-	Use value present in the default mask; do not override. Used in override masks only.
X	Don't care. Value unused by guest software. Not checked for VMotion.
T	Bit (feature) must be set in hardware and is required by software in the virtual machine. May be used in default or override masks. Bit must match (set) for VMware VMotion.
F	Bit (feature) must not be set in hardware, hence feature must not be enabled for software running in virtual machine. May be used in default or override masks. Bit must match (clear) for VMotion.
1	Set this bit if `CPUID` is executed by guest software. May be used in default or override masks. Not checked for VMotion.
0	Clear this bit if `CPUID` is executed by software in the virtual machine. This is used to hide feature from software in the virtual machine. May be used in default or override masks. Not checked for VMotion.
H	Allow software in the virtual machine to see actual feature value returned by `CPUID` on hardware. Likely to be used in default masks but may be used in override masks as well. Must match for VMotion.
R	Hide feature from software in the virtual machine by clearing this bit. Likely to be used on default masks but may be used in override masks as well. Must match for VMotion.

Default Masks

Starting with VirtualCenter 2, default masks are defined for each guest operating system. For VMotion checks, even though the R entries are hidden from the software in the virtual machine, they are required to match, because application-level software may be utilizing the features by querying `CPUID` directly on hardware (without enhanced VMotion support). The H values are required to match for VMotion, because software in the virtual machine utilizes these features in hardware. The T values indicate the guest operating system cannot function without these features, therefore the features must be set in hardware (for example, the long mode bit CPUID level 0x80000001h EDX:29 for 64-bit guest operating systems). This implies that the virtual machine requires this feature even on the new destination host hardware. Therefore, this value is required for VMotion. Similarly, the F values indicate a guest cannot function if this feature is present, therefore the feature must not be present on the destination hardware when performing VMotion.

The default masks for a virtual machine are stored in an XML file named `vmconfigoption-esx-<version>.xml`, where `<version>` is the three-digit version of ESX—for example, `3.0.0`. This XML file is located in the `/etc/vmware/hostd/env` directory.

Override Masks

An override mask replaces the existing default mask value for a particular CPUID bit. Override masks have an implicit empty value (-), which means there is no override value for that bit. Override masks are per virtual machine and are stored in the virtual machine's configuration file (`.vmx`). To override a particular CPUID bit, you must enter override values for the entire register for the corresponding CPUID level. For instance, to disable the SSSE3 feature for software in the virtual machine, clear CPUID level 0x1h ECX:9. The override mask to accomplish this is:

```
Level 0x1h
ECX: ---- ---- ---- ---- ---- --0- ---- ----
```

This override mask disables only the SSSE3 bit in CPUID level 0x1h ECX:9 while preserving all other default values that VirtualCenter sets for this combination of register and level.

NOTE You can create overrides only when the virtual machine is powered off, and the override mask takes effect only when the machine is started.

Final Mask

VirtualCenter calculates the final mask by overriding values in the default mask using the values in the override mask, if there is one. This final mask then becomes part of the VirtualCenter configuration and is applied to the results of the `CPUID` instruction.

Continuing with the SSSE3 example above, assume the default mask for CPUID level 0x1h ECX is:

```
Level 0x1h
ECX: RRRR RRRR RRRR RRR0 00xR R0H0 000H 0RRH
```

You then apply the following override mask:

```
Level 0x1h
ECX: ---- ---- ---- ---- ---- --0- ---- ----
```

This gives you a final mask for CPUID Level 0x1h register ECX of:

```
Level 0x1h
ECX: RRRR RRRR RRRR RRR0 00xR R000 000H 0RRH
```

To test VMotion compatibility, VirtualCenter applies this mask to the results of the `CPUID` instruction on the source and destination hardware and compares all registers at various levels. CPU compatibility checks match only if all bits match after the mask is applied.

Appendix C: Relaxing VMware VMotion CPU Compatibility Constraints

If you need to use VMotion between hosts with different CPUs in environments that do not support Enhanced VMotion Compatibility clusters, you may find it useful to follow the guidelines in this appendix. If your environment allows you to use EVC clusters, you do not need to use the techniques described in this appendix.

VirtualCenter checks to ensure that the source and target CPUs are compatible before allowing migration with VMotion. It does so by ensuring feature and extended feature flags match. It is possible to circumvent these checks by applying override masks. These override masks disable certain features for guest operating systems on the host and destination CPUs and therefore allow VirtualCenter to complete VMotion checks successfully.

Assuming the guest operating system uses the CPUID instruction to detect features supported by a CPU, the features masked out are not available to the guest operating system software. The virtual machine must be restarted so application software in the virtual machine can run feature detection (via the CPUID instruction) and use only the set of features prescribed by the override masks.

NOTE These relaxations are not supported.

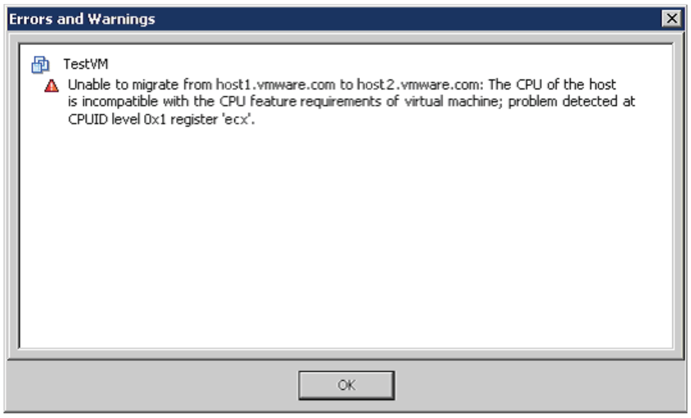
The following example shows how to disable SSE4.1 instructions on an Intel CPU so that a virtual machine can be migrated to a host with a CPU that lacks this feature.

NOTE This feature is not supported because application software that relies on running the CPUID instruction for feature detection directly on hardware can detect this feature and might rely on it. In an environment that supports EVC, you do not need to be concerned with the masking described in this appendix.

The example uses the following hosts and CPUs:

- host1.vmware.com: two-core Intel Penryn CPU
- host2.vmware.com: four-core Intel Kentsfield CPU

Assume all compatibility checks for VMotion other than the CPU check complete successfully. While performing VMotion from host host1.vmware.com to host host2.vmware.com on VMware Infrastructure 3, the following message appears:



TestVM
Unable to migrate from host1.vmware.com to host2.vmware.com: The CPU of the host is incompatible with the CPU feature requirements of virtual machine; problem detected at CPUID level 0x1 register 'ecx'.

This message indicates that the CPU compatibility checks performed prior to migration failed. The CPU in host1.vmware.com possesses SSE4.1 extensions, but the CPU in host2.vmware.com does not. You can determine these CPU features by executing the CPUID instruction on these hosts. You can obtain the exact value of the mismatched bits by running the CPU information utility (cpuid.iso) provided with VMware Infrastructure 3. You can uncompress the ISO image file (\images\cpuid.iso.gz) and use it to create a bootable CD-ROM that provides CPU information about a given host before you install an operating system or ESX. Running this utility on the source host (host1.vmware.com) gives the value of registers when CPUID is executed at various levels. You can use it to examine the cause of the mismatch.

Run the CPUID image on host1.vmware.com (Intel Penryn CPU). The output includes the following relevant information:

```
Family: 06 Model: 17 Stepping: 4

ID1ECX     ID1EDX     ID81ECX    ID81EDX
0x0008e3bd 0xbfebfbff 0x00000001 0x20100800
```

Run the CPUID image on host2.vmware.com (Intel Kentsfield CPU). The output includes the following relevant information:

```
Family: 06 Model: 0f Stepping: 7

ID1ECX     ID1EDX     ID81ECX    ID81EDX
0x0000e3bd 0xbfebfbff 0x00000001 0x20100800
```

The compatibility error message indicates that CPUID level 1 ECX bits do not match. (VirtualCenter checks for compatible families only and ignores model and stepping information.) An alternative method of determining features present on a host CPU is using the managed object browser (MOB) present in ESX. You can access CPUID bits for all levels using the MOB. The values of registers are indicated in binary format. The level numbers are reported in integer format.

Look more closely at ID1ECX on the two machines.

CPUID level 0x1h ECX (host1.vmware.com) 0x0008e3bd

31				27				23				19				15				11				7				3			0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	1	1	1	0	1	1	1	1	0	1

CPUID level 0x1h ECX (host2.vmware.com) 0x0000e3bd

31				27				23				19				15				11				7				3			0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	1	1	0	1	1	1	1	0	1

The registers differ in the value of bit 19, which indicates support for the SSE4.1 set of instructions. You can also obtain the bits differing between the two registers by using a bit-wise XOR function.

Look at the default masks in VirtualCenter 2.0. They are set to require a match for the value of CPUID level 1 ECX:19 between source and destination. To circumvent this check, you must:

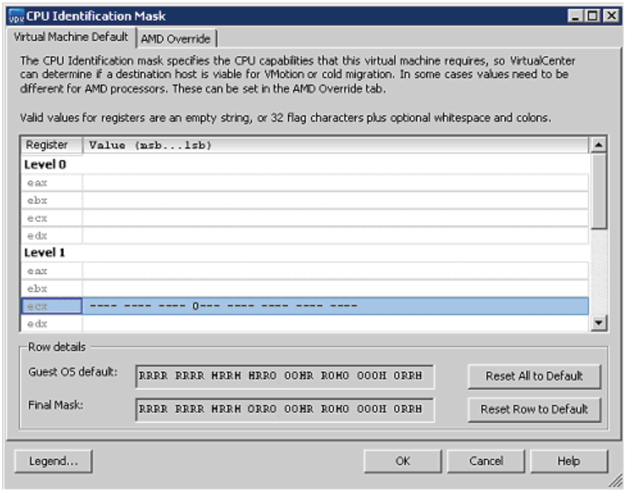
1. Apply appropriate masks to hide this feature from the virtual machine.
2. Establish that no application requires the feature you masked.
3. Reboot the virtual machine.

The override mask for each virtual machine is stored in its configuration (`.vmx`) file.

To mask CPUID level 0x1h ECX:19, apply the following mask in VirtualCenter.

31				27				23				19				15				11				7				3			0
-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

The mask `---- ---- ---- 0--- ---- ---- ---- ----` applied to level 1 ECX hides the SSE4.1 feature from software in the virtual machine.



CPU Identification Mask

Virtual Machine Default | AMD Override

The CPU Identification mask specifies the CPU capabilities that this virtual machine requires, so VirtualCenter can determine if a destination host is viable for VMotion or cold migration. In some cases values need to be different for AMD processors. These can be set in the AMD Override tab.

Valid values for registers are an empty string, or 32 flag characters plus optional whitespace and colons.

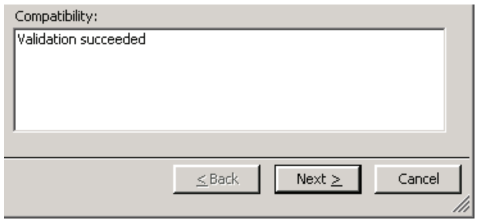
Register	Value (msb...lsb)
Level 0	
eax	
ebx	
ecx	
edx	
Level 1	
eax	
ebx	
ecx	`---- ---- ---- 0--- ---- ---- ---- ----`
edx	

Row details

Guest OS default: `RRRR RRRR HRRH HRRO OOHR ROHO OOOH ORRH`

Final Mask: `RRRR RRRR HRRH ORRO OOHR ROHO OOOH ORRH`

For the new masks to take effect, you must restart the virtual machine. When the new mask is in effect, migration from host1.vmware.com to host2.vmware.com should succeed.


Compatibility:

Validation succeeded

See the VMware knowledge base article "VMotion CPU Compatibility —Migrations Prevented Due to CPU Mismatch—How to Override Masks" for more examples showing how other features can be masked. See "References" on page 7 for a link.