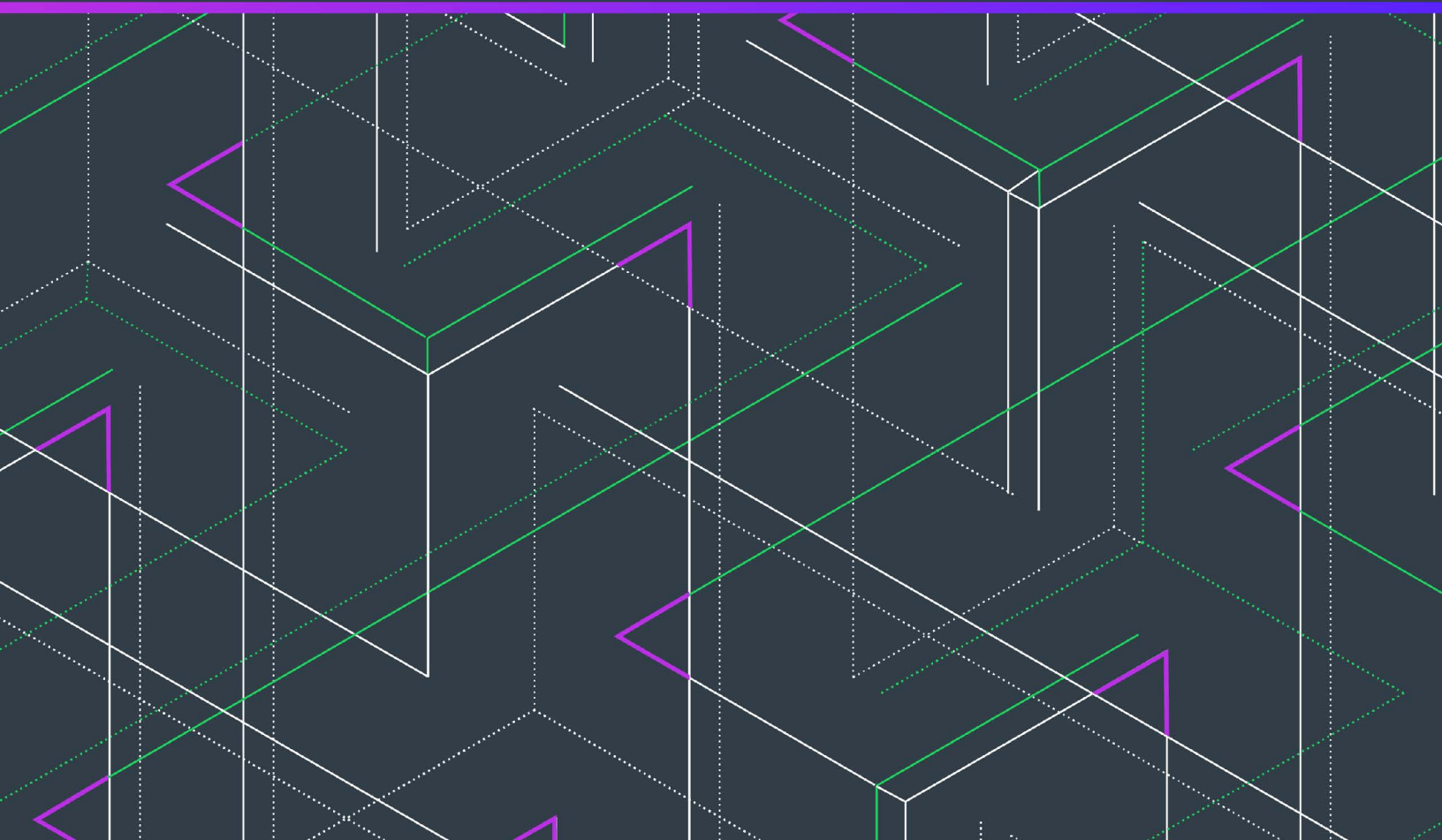


FlexNet Publisher

White Paper: Understanding Virtualization Features in
FlexNet Publisher



Legal Information

Book Name: White Paper: Understanding Virtualization Features in FlexNet Publisher
Part Number: FNP-WP-UVF-2312
Last Updated: December 2023

Copyright Notice

Copyright © 2023 Flexera Software

This publication contains proprietary and confidential information and creative works owned by Flexera Software and its licensors, if any. Any use, copying, publication, distribution, display, modification, or transmission of such publication in whole or in part in any form or by any means without the prior express written permission of Flexera Software is strictly prohibited. Except where expressly provided by Flexera Software in writing, possession of this publication shall not be construed to confer any license or rights under any Flexera Software intellectual property rights, whether by estoppel, implication, or otherwise.

All copies of the technology and related information, if allowed by Flexera Software, must display this notice of copyright and ownership in full.

FlexNet Publisher incorporates software developed by others and redistributed according to license agreements. Copyright notices and licenses for these external libraries are provided in a supplementary document that accompanies this one.

Intellectual Property

For a list of trademarks and patents that are owned by Flexera Software, see <https://www.reverera.com/legal/intellectual-property.html>. All other brand and product names mentioned in Flexera Software products, product documentation, and marketing materials are the trademarks and registered trademarks of their respective owners.

Restricted Rights Legend

The Software is commercial computer software. If the user or licensee of the Software is an agency, department, or other entity of the United States Government, the use, duplication, reproduction, release, modification, disclosure, or transfer of the Software, or any related documentation of any kind, including technical data and manuals, is restricted by a license agreement or by the terms of this Agreement in accordance with Federal Acquisition Regulation 12.212 for civilian purposes and Defense Federal Acquisition Regulation Supplement 227.7202 for military purposes. The Software was developed fully at private expense. All other use is prohibited.

Contents

Introduction	5
Abbreviations Used in This Document	5
Virtualization Taxonomy	5
Virtualization Problem Statement	7
Virtualization Stack Support in FlexNet Publisher	8
Historical Virtualization Support in FlexNet Publisher	9
Virtual Machine HostIDs	11
VMID	12
MAC address	12
VM GenerationID	12
HostID behavior during VMM-managed VM transition events	14
VMware Workstation and VM HostIDs	15
FNP-TS Behavior in Virtual Environments	15
Virtualization-aware binding	15
Cloud-aware binding	16
Recovering from a binding break in TS	16
Live snapshot of a TS-based Windows license server	18
The FlexNet Licensing Service and Virtualization Detection	18
License File-Based Licensing and VM GenerationID	19
Summary of Recommended Virtualization Functionality in FlexNet Publisher from 11.15.0	20
Deprecated functionality	20
FNP-Certificate	20
Server HostID	20
Client HostID	21
Dongles and USB passthrough	21
FNP-TS	22
Troubleshooting FNP-TS Behavior in Virtual Environments	22

Appendix: Generating a Vendor Daemon Nag Message 25

Process Flow for Generating and Using License Files..... 25

Adapting Isvendor.c to Generate a Nag Message when GenerationID Changes..... 26

Introduction

This White Paper describes virtualization challenges in software licensing and the facilities available in FlexNet Publisher for operating licensing models in a virtualized environment.

This White Paper refers to FlexNet Publisher 2018 R1 (11.15.1) and later.

Abbreviations Used in This Document

Table 1 • List of abbreviations

Abbreviation	Description
FNP	FlexNet Publisher
FNP-TS	FlexNet Publisher with Trusted Storage
TS	Trusted storage
UMN	Unique machine number
VMID	Virtual machine identifier, from which VM_UUID and UMN3 are derived
FNLS	FlexNet Licensing Service

Virtualization Taxonomy

In this section we categorise the terms: hypervisor, virtual machine (VM), cloud virtual machine (CVM), VM transition event, virtual machine manager (VMM) and virtualization stack.

A *hypervisor* manages virtual machines on a native host.

A *virtual machine* (or VM, or Guest) is an instance of a virtualized OS together with a set of virtualized hardware resources.

A *cloud virtual machine* (CVM) refers to a virtual machine instance created in a third-party-managed IaaS or SaaS environment, such as Amazon EC2, Microsoft Azure or Google Compute.

A *virtual machine manager* (VMM) maintains one or more hypervisor instances, and provides the interface for managing VM transition events. Client VMMs are intended to be lightweight hypervisor administrative tools which may reside on multiple distributed machines (such as the author's notebook), whereas server VMMs are intended to provide centralized management of a virtualized data center.

Table 2 • Virtualization taxonomy

Hypervisor	VMM Client	VMM Server
VMware Workstation	VMware Workstation	N/A

Table 2 • Virtualization taxonomy

Hypervisor	VMM Client	VMM Server
VMware ESXi	VMware vSphere client	vCenter Server
Hyper-V	Hyper-V Manager	System Center 2016
Citrix XenServer	Citrix XenCenter	Citrix XenCenter
Nutanix	Prism Element	CVM (Controller virtual machine)

VM transition events fall into one of three categories: clone, snapshot or migrate. All properly-managed VM transition events occur by means of a VMM.

A *clone* provides a copy of a VM either on the same or a different host. For the purposes of this licensing discussion, we treat the Backup/Restore, Export/Import, copy (of VM folder), copy from template and Upload/Download transition events as variations of clone, because the licensing leakage impact is identical to that of cloning.

A *snapshot* occurs when a backup of a VM's state is taken at a point in time. A revert-to-snapshot reverts a VM to its earlier state. Cold snapshots include only disk state; live snapshots additionally include memory state.

A *VM migration* moves the VM from one host to another. Migrations can be cold (offline, when the VM is shut down) or live, meaning the memory state is migrated as well.

Hypervisors can be categorized in multiple ways. A common differentiation is Type 1 (bare metal) versus Type 2 (hosted on a native OS). Type 2 hypervisors include VMware Workstation, Oracle VirtualBox and Parallels. Type 1 hypervisors can be further differentiated according to whether they operate side-by-side with a standard OS (Hyper-V, QEMU-KVM), whether they have a publically available but stripped-down OS as part of the hypervisor (VMware ESXi, Citrix XenServer, and Nutanix). More generally, it is useful to ask if the Type 1 hypervisor provider supports running third-party applications on the host, since this provides an opportunity for running bare-metal licensing components. Another hypervisor differentiator is to ask whether one might expect to see the hypervisor primarily deployed on server machines or on desktops. This has implications for licensing; for example, a hypervisor in a data centre is less likely to be subject to adversarial experimentation because of the stability and compliance implications it would have in the data center. A desktop-based hypervisor, however, can be assumed to exist in a more hostile environment.

Table 3 • Hypervisors

Hypervisor	Hypervisor Category	Server / Desktop
VMware ESXi	Bare Metal: Closed OS	Server
VMware Workstation	Hosted	Desktop
Hyper-V with Windows Server	Bare Metal: Side-by-side OS	Server
Hyper-V with Windows 10	Bare Metal: Side-by-side OS	Desktop
Citrix XenServer	Bare Metal: Closed OS	Server

Table 3 • Hypervisors

Hypervisor	Hypervisor Category	Server / Desktop
QEMU-KVM	Bare Metal: Side-by-side OS	Server
Oracle VirtualBox	Hosted	Desktop
Parallels	Hosted	Desktop
Nutanix	Bare Metal: Closed OS	Server

A *virtualization stack* defines the virtualization environment and has the following four levels: Hypervisor OS (or Host OS); Hypervisor; Guest OS; and the FNP kit used in the Guest OS. Examples of virtualization stacks include

- Windows Server 2016; Hyper-V; Windows 10; x64_n6 FNP.
- Windows 10; VMware Workstation 12; Red Hat Enterprise Linux 6; x64_linux FNP
- ESXi 6.5 (Host OS and Hypervisor); Red Hat Enterprise Linux 7; x64_linux FNP

Virtualization Problem Statement

When virtual machines are cloned, licensing leakage can occur. These license leakage scenarios are most acute for license servers, because servers can serve multiple licenses.

Particularly important is the challenge of *accidental* leakage. An example is when a system administrator accidentally performs a VM clone to a new host instead of a migrate.

Snapshots present additional license leakage opportunities, particularly for TS-based licensing. Restoring a VM to a snapshot can restore TS to a more favourable licensing state. Snapshot-restore may also occur in a 'fair-use' manner, for example if a VM exhibits instability a system administrator may revert it to an earlier known-stable snapshot.

Balancing the need to prevent license leakage is the requirement not to break licensing when migrating VMs. A VM migration is not a license leakage scenario. VM migration, particularly live migration (VMware call this VMotion), is used in high-availability environments.

Finally, licensed applications can be characterized as existing in normally connected or normally disconnected environments. A normally connected environment provides opportunities for license servers and clients to frequently synchronise with a point of trust external to the enterprise, such as Revenera's FNO Cloud. This provides a number of potential mechanisms for dealing with license leakage, such as a short-lived SessionID held between the license server and the point of trust in the cloud. However, FNP servers and clients exist in normally disconnected environments, meaning only occasional access to an external point of trust (such as the FNO backoffice) can be assumed. This limits the range of solutions available for addressing license leakage.

Virtualization Stack Support in FlexNet Publisher

FlexNet Publisher restricts its Guest support to x86 and x86_64 based Linux and Windows operating systems. The versions of Windows and Linux supported as Guests can be found in FlexNet Publisher *Release Notes*. Only the following four FNP kits support virtualization features: i86_n3; x64_n6; i86_linux; x64_linux.

There are three levels of cumulative hypervisor support in FNP

1. Unaware and unbroken
2. Binary virtualization detection
3. HostID extraction

Level-1 support says FNP will behave in a virtualization unaware fashion when in a Guest of an undetected hypervisor. This means it will treat the Guest as a physical system and can be expected to work exactly as if it were on a physical system.

Level-2 support allows binary detection of FNP being in some virtual environment, without identifying that environment. `lc_virtualstatusget` is a binary virtualization detection API provided by FNP.

Level-3 support is where one or more suitable HostIDs made available by the hypervisor to the Guest are extracted by FNP when running in the guest. These HostIDs can be used in cloning mitigation and in licensing lifecycle operations.

Currently FNP offers Level-3 support for

- these hypervisors: VMware ESXi, VMware Workstation, Hyper-V, Citrix XenServer, Oracle VirtualBox, Parallels, QEMU-KVM, Stratus everRun, and Nutanix.
- CVMs running on these third-party cloud environments: Amazon EC2, Microsoft Azure and Google Compute

OpenStack is not explicitly tested by FNP. However, since OpenStack provides the Amazon EC2 metadata interface, OpenStack CVMs will currently be detected as Amazon EC2 instances by FNP.

Historical Virtualization Support in FlexNet Publisher

Table 4 • Historical virtualization support

FNP version	Feature	Details	Hypervisors Supported	Limitations
11.7	Detect-and-deny	<ul style="list-style-type: none"> ls_allow_vm vendor variable VM_PLATFORMS license file keyword 	VMware ESX & Workstation	No trusted storage support
11.8	License server cloning mitigation	<ul style="list-style-type: none"> SERVER line HostID support : VMW_UUID, other VMW_* such as VMW_ETHER Physical hostid support: PHY_* (example - PHY_ETHER). Physical HostIDs require a physical machine. Lmbind. LMB_* hostids (later removed) 	VMware ESX & Workstation	<ul style="list-style-type: none"> No trusted storage support Imbind supported only on the host machine cloning mitigation only for license servers
11.9	Hyper-V support	Equivalent support to VMware	VMware ESX & Workstation Hyper-V with Windows Server	No trusted storage support
11.10	TS support with VMID binding	<ul style="list-style-type: none"> flxActCommonVirtualStatusGet() Activation API Detect-and-deny locally activated trials (VIRTUALIZATION_POLICY ASR keyword) bind-to-VMID policy Using VMID (UMN3) in licensing lifecycle operations 	VMware ESX & Workstation Hyper-V with Windows Server	No Imbind support with TS

Table 4 • Historical virtualization support

FNP version	Feature	Details	Hypervisors Supported	Limitations
11.10	Amazon EC2 support	<ul style="list-style-type: none"> Support certificate license servers running in EC2 instances AMZN_IID, AMZN_EIP, AMZN_AMI SERVER line hostids 	Amazon EC2	No trusted storage support
11.11	Citrix XenServer support	<ul style="list-style-type: none"> Parity support with other hypervisors VM_UUID SERVER-line HostID generically available on VMware, Hyper-V and Xen 	VMware ESX, ESXi & Workstation Hyper-V with Windows Server Citrix XenServer	Imbind not supported on Xen's closed OS
11.12.1	Enhanced cloning mitigation in TS	Environment-aware binding and use of VM GenerationID in binding.	VMware ESXi & Workstation Hyper-V with Windows Server and Windows 8.x Citrix XenServer	Enhanced cloning mitigation limited to TS licensing models
11.13.0	Oracle VirtualBox support	VM_UUID extraction supported on VirtualBox.	As in 11.12.1 + VirtualBox	Oracle VirtualBox does not support GenerationID
	Enhanced cloning mitigation in license-file-based licensing	Support for extracting VM GenerationID in license-file-based applications	As in 11.12.1	
11.13.1	QEMU-KVM & Parallels Desktop support	Parallels runs on OS X	As in 11.13.0 + QEMU-KVM + Parallels Desktop	Parallels does not change VMID on VM clone
	Detection differentiation between VMware Workstation and ESXi			

Table 4 • Historical virtualization support

FNP version	Feature	Details	Hypervisors Supported	Limitations
11.14.0	TS support introduced on Amazon EC2 TS and certificate support introduced for Microsoft Azure and Google Compute	Detection for supported third-party cloud environments introduced. Cloud-aware binding for TS introduced VM_UUID can be used as server line HostID on all supported third-party cloud environments	As in 11.13.1	Azure detection from a Linux guest requires the Linux FNLS
11.14.1	Fast virtualization detection introduced		As in 11.13.1	Virtualization detection completely delegated to the FNLS. If FNLS not installed, all machines are treated as physical.
11.15.0	Fast binary virtualization detection in absence of FNLS introduced	Virtualization detection now available in absence of FNLS.	As in 11.13.1	FNLS still required for determining hypervisor or cloud type, and for extraction of VM_UUID or AMZN_EIP.
11.18.3	The detection technique for VM_UUID in Azure environment changed	It provides security, stability, and ease to use for standard users. This technique is similar to other FlexNet Publisher cloud solutions.	As in 11.13.1	The new detection technique will generate different VM_UUID value. The FNP artifacts like license file and trusted storage may require modification to the latest VM_UUID value.
11.19.3	Nutanix support	VM_UUID and ETHERNET extraction supported on Nutanix	As in 11.19.3+	Nutanix does not support GenerationID.

Virtual Machine HostIDs

A Virtual Machine HostID is a property of the VM itself, and this property is managed by the hypervisor.

Effectiveness of HostIDs for virtual machines can be evaluated according to the following criteria.

- Does the HostID change when a VM is cloned using the VMM?
- Does the HostID change when the VM is reverted to a snapshot?
- Does the HostID remain constant when a VM is migrated?
- How difficult is it to spoof the HostID?

FNPN considers three types of HostIDs that can be extracted from a VM: MAC address (of a virtualized network interface card); VMID and GenerationID. Of the three, only GenerationID is effective against revert-to-snapshot.

VMID

The VMID refers to the VM UUID. This is the virtualized SMBIOS UUID on hypervisors and Azure, or the CVM Instance ID on Amazon EC2 or Google Compute. An exception is Linux Guests on Xen, where Xen may synthesize the VMID. SMBIOS UUID as VMID was introduced by VMware; this is the 'uuid.bios' value found in VMware configuration files. Certificate license servers use VMID via the VM_UUID keyword on the SERVER line. FNP-TS uses VMID in binding, and the VMID is sent to the backoffice in activation requests as UMN3 (see the Revenera white paper *Understanding the UMN in FlexNet Publisher*). VMID is a solid HostID on VMware, Nutanix, and Xen, where a clone will change the value, but not on Hyper-V where cloning does not change VMID.

MAC address

MAC address is a solid all-round HostID on virtual machines. Hypervisors will generally change MAC address on clone, but not on a migrate, as desired. While it can be spoofed relatively easily, for example via most VMM GUIs, VMMs tend not to allow multiple VMs with the same MAC address, because MAC address collisions can occur if two machines with the same MAC address run on the same subnet.

VM GenerationID

The newest HostID is VM GenerationID. VM GenerationID was introduced by Microsoft. It was invented in order to solve issues occurring in Active Directory as a result of VM transition events such as clone and snapshot. GenerationID is a 128-bit value provided by the hypervisor to its VMs via the ACPI namespace of the virtual machine.

The application of VM GenerationID to software licensing follows from the Microsoft-published behavior (see [https://msdn.microsoft.com/en-us/library/jj643357\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/jj643357(v=vs.85).aspx)):

Table 5 • Microsoft-specified VM GenerationID behaviour

Scenario	Generation ID changed
Virtual machine is paused or resumed	No
Virtual machine reboots	No
Virtual machine host reboots	No
Virtual machine starts executing a snapshot (every time)	Yes
Virtual machine is recovered from backup	Yes
Virtual machine is failed over in a disaster recovery environment	Yes
Virtual machine is live migrated	No

Table 5 • Microsoft-specified VM GenerationID behaviour

Scenario	Generation ID changed
Virtual machine is imported, copied, or cloned	Yes
Virtual machine is failed over in a clustered environment	No

As can be seen from the above table, the behavior of GenerationID maps exactly to what is desired in a licensing environment, apart from the disaster recovery scenario.

GenerationID is generally harder to spoof than VMID or MAC address in virtual environments. The advantages of Generation ID over the other two HostIDs are therefore

- harder to spoof
- the only HostID that changes on revert-to-snapshot

GenerationID has to be supported by both the hypervisor and by a native ACPI driver in the Guest OS. Microsoft provides this driver natively in Windows 8+ and Windows Server 2012+. The following table lists relevant hypervisor & Guest OS combinations.

Table 6 • VM GenerationID support by Hypervisor and Guest OS

Hypervisor	Guest	VM GenerationID?
Hyper-V (with Windows Server 2012+ or Windows 8+)	Windows 7, with Hyper-V Integrated Services installed	Yes
	Windows Server 2008, with Hyper-V Integrated Services installed	Yes
	Windows Server 2012+	Yes
	Windows 8.1, 10	Yes
	Any Linux	No
VMware (ESXi 5.1+ or Workstation 9+) or Citrix XenServer 6.2+ or everRun 7.7+	Windows 7	No
	Windows Server 2008	No
	Windows Server 2012+	Yes
	Windows 8.1, 10	Yes
	Any Linux	No
Oracle VirtualBox Parallels QEMU-KVM	Any	No

Table 6 • VM GenerationID support by Hypervisor and Guest OS

Hypervisor	Guest	VM GenerationID?
Nutanix	Windows	No
	Linux	No

As can be seen, Linux guests are not currently supported.

HostID behavior during VMM-managed VM transition events

Tested hypervisors include: VMware ESXi 6.5, VMware Workstation 12, Hyper-V with Windows Server 2012 R2 & 2016, Hyper-V with Windows 10 and Citrix XenServer 6.5 & 7.0.

Tested guest OSs include: Windows Server 2012 R2, Windows 10, Windows 7 SP1 and Red Hat Enterprise Linux 6.

Tested VMMs include: VMware vSphere Client, VMware vCenter Server, Hyper-V Manager, Microsoft System Center 2012 and Citrix XenCenter.

The table below indicates whether the HostID changed as a result of the VM transition event.

Table 7 • Tested VM HostID behaviour

VM Transition Event	VMID (VMware, XenServer, QEMU-KVM)	VMID (Hyper-V & Parallels)	MAC address	GenerationID (Windows Guest)
Clone	Yes	No	Usually ¹	Yes
Clone from template	Yes	No	Usually ¹	Yes
Imported from export ²	Yes	No	Usually ¹	Yes
Restored from backup	Yes	No	Usually ¹	Yes
Cold or Live migrated	No	No	No	No
Restored to Snapshot	No	No	No	Yes



Note • MAC address change after clone will occur

- If source and clone VM are managed by the same VMM instance
- Normally at least on first start of a cloned VM
- Sometimes only on second start of a cloned VM

- Sometimes only after resolution of MAC address collision: If source and clone are started in the same subnet, this can manifest as one of them not being able to get network access.



Note ▪ VMware does not support Import/Export



Note ▪ In everRun environment, only the clone transition event is tested in this release. The results will be same as in XenServer environment.

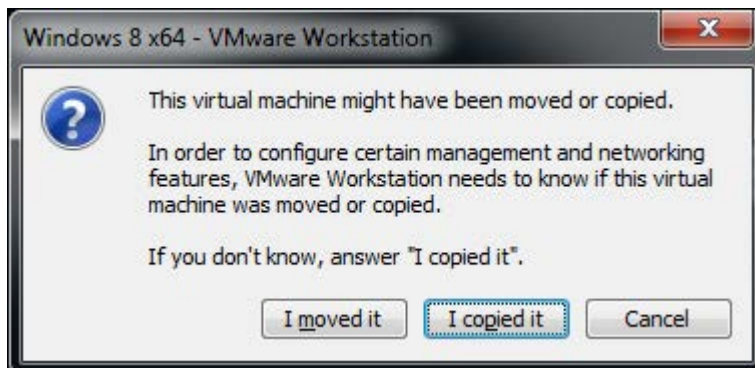


Note ▪ The clone transition event is tested in the Nutanix environment, leads to VMID and MAC address modifications.

VMware Workstation and VM HostIDs

Workstation VMs can be cloned independent of a VMM by copying the VM folder on disk, without using the Workstation GUI.

If a Workstation VM folder on disk is copied and the copied vmx file opened in Workstation, VMware asks the following question:



Answering 'I copied it' (the default) changes the VMID and VM GenerationID HostIDs, whereas 'I moved it' keeps all HostIDs constant. 'I moved it' is intended to be the equivalent of cold migration, whereas 'I copied it' is a variation of clone. If the end-user clicks 'I moved it', and did not delete the original folder, this can result in license leakage. However, if the source and cloned VM are started in the same subnet, MAC address collisions can occur.

FNP-TS Behavior in Virtual Environments

Virtualization-aware binding

From 11.12.1, FNP binding behaves in an environment-aware fashion on VMs: in physical systems, the original flexible binding is used, whereas in virtual systems, FNP binds to a different set of binding items and with more stringent binding rules.

FNP now introduces the concept of dealbreaker binding. If any one dealbreaker binding item changes, then TS becomes untrusted. This is in contrast to traditional flexible binding where more than 50% of a weighted average of all binding items needs to change all at once in order for TS to become untrusted. The following binding items are all dealbreakers in TS virtual environments (no other items are used in virtual binding):

- VMID
- GenerationID, if present.
- The set of MAC addresses which the Guest OS reports as being attributed to physical NICs.



Note - Most Guest OSs will 'see' a virtualized NIC allocated to it by a hypervisor as a physical NIC.

In the case of MAC address, there can be more than one MAC address available to a machine. For example, hypervisors support allocating multiple virtualized NICs to a guest OS. FNP binding requires the entire set of all MAC addresses to change at once in order for TS to become untrusted. This is in order to provide a balance between normal system administration of a VM and countering cloning exploits.

From FNP 11.12.1, specifying a bind-to-VMID configuration in FNO is unnecessary, since FNP will automatically choose the binding style (cloud versus virtual versus flexible) according to the environment. In FNP 11.12.1, a bind-to-VMID configuration will automatically be upgraded to virtual-environment binding. Unlike bind-to-VMID, virtual license servers with virtual-environment binding can support activation-borrow to physical or virtual clients.

Cloud-aware binding

From 11.14.0, FNP binding behaves in an environment-aware fashion on CVMs: in supported third-party cloud environments, TS binds to just one binding item - the VMID of the cloud environment.

VM GenerationID is not used for binding in cloud environments, because it is not reliably generated.

Recovering from a binding break in TS

This discussion assumes FNP 11.12.1+ in virtual environments.

The TS model is that a broken TS can be recovered by a transaction with the backoffice, normally with a repair, or by means of a reinstall. If a repair or reinstall fails, a new license must be consumed by means of a new activation.

However, since a clone of a VM typically changes some or all of the previously discovered HostIDs, FNO will in general deny repair or reinstall requests originating from cloned VMs.

Common scenarios leading to broken TS (assuming a Windows guest with VM GenerationID supported) are detailed in the following table.

Table 8 ▪ Scenarios leading to broken TS

Event leading to broken TS (Windows guest)	Repair / Reinstall possible with FNO?	<BreakInfo> in repair request ¹	UMNs changed	UMNs unchanged	Comments
Clone (Hyper-V)	No	ETHERNET GENID	UMN2 UMN5	UMN3	In some cases, such as clone to host on different subnet, UMN2/ETHERNET may not change, in which case repair is possible.
Clone (VMware)	No	VMID GENID	UMN3 UMN5	UMN2	User may be forced to change MAC address (causing additional ETHERNET break) in order to resolve MAC collision and/or network issues
Clone (Xen)	No	ETHERNET VMID GENID	UMN2 UMN3 UMN5	None	
Revert-to-Snapshot	Yes	GENID	UMN5	UMN2 UMN3	In FNO 12.11+, requires Deny Repair if VM Generation ID is changed set to No
System administrator changes MAC address of VM via VMM interface	No	ETHERNET	UMN2	UMN3 UMN5 ²	Assumes only one MAC address was available to the OS. If there is more than one MAC address, all addresses need to be changed before the next TS read in order to cause a break.



Note ▪ As generated by a command such as: `appcomptranutil -new c:\test\repair.xml -repairAll`



Note ▪ As of FNO 12.11, UMN5 is ignored by FNO.

Linux guests will show similar behavior, except for the following

- GENID will not be present in BreakInfo, nor will UMN5 be present
- Linux guests will not incur a TS break on revert-to-snapshot

Live snapshot of a TS-based Windows license server

A live snapshot takes a snapshot of memory as well as disk. Hyper-V is an example of a hypervisor that allows live snapshotting. This offers potential for an unlimited exploit of licenses, for example:

1. Start license server with 10 hybrid licenses.
2. Take live snapshot.
3. 10 clients borrow-activate all the licenses.
4. Revert-to-live-snapshot. Because the VD is not restarted, TS binding and therefore GenerationID is not updated.
5. A further 10 clients borrow-activate licenses.
6. Repeat from step 4.

From FlexNet Publisher 11.12.1, TS-based license servers poll TS every 90 seconds to determine if there has been a binding break. Currently, TS will incur such a 'live break' only if GenerationID changes. Hence, within at least two minutes of a GenerationID-aware TS-based license server being reverted, it will shut down, with server log messages similar to:

```
12:00:46 (demo) Trusted storage binding change detected! Vendor daemon is being shutdown.
12:00:46 (demo) EXITING DUE TO SIGNAL 65 Exit reason 42
12:00:51 (lmgrd) demo exited with status 65 (Trusted storage binding change detected.)
12:00:51 (lmgrd) Vendor daemon shutdown - trusted storage binding change detected
```

The FlexNet Licensing Service and Virtualization Detection

FNPN extracts all possible virtualization information in a set; we call this information the VM attributes. VM attributes comprise identification of the hypervisor (and cloud environment if in a CVM) as well as the VMID and (if present) the VM Generation ID and (if on an EC2 CVM) AMZN_EIP. There are two reasons why extraction of VM attributes have, since 11.14.1, been delegated to the FNLS:

- Extracting VM attributes can take of the order of 100 milliseconds; when querying a CVM's metadata interface in some environments (for example, some OpenStack implementations) it can take even longer. In order not to slow down the first checkout of FNPN clients, VM attributes are cached—with periodic refresh—in the FNLS.
- Some VM attributes require elevated/root privilege to extract - for example: VM Generation ID and detection of Azure from a Linux CVM.

From 11.15.0, level 2 virtualization support is offered without the FNLS. This is because a binary yes/no answer for virtualization is relatively fast and does not need elevated privilege to determine. FNLS continues to be required for the extraction of all other VM attributes.

License File–Based Licensing and VM GenerationID

Starting from 11.12.1, FNP supports extraction of the VM GenerationID via one of

- `lc_hostid(job, VM_GENID, buffer)`. This can be called by clients and in license server callback APIs.
- `lmhostid -ptype VM -genid`
- the new `lmvminfo` utility (`lmvminfo -long`)

Extracting GenerationID incurs a dependency on the FlexNet Licensing Service, because elevated privileges are required to query the ACPI driver.

The following discussion focusses on license servers:

Because of concerns that issuing a "VM_GENID" license file keyword would result in overly brittle licenses, Revenera has not delivered such a keyword. Instead the recommendation is to issue a separate license file (call it the **GID license**) with GenerationID in the `VENDOR_STRING` of a dummy feature. Then have the vendor daemon poll for GenerationID changes and issue a nag message, or log the fact if it does change. [Appendix: Generating a Vendor Daemon Nag Message](#) gives a detailed example of how to do this for the vendor daemon using the `ls_daemon_periodic` callback function in `lsvendor.c`.

If we consider the system administrator described in the problem statement, such an admin may accidentally clone a virtual machine (instead of migrating it), or revert it to a known stable snapshot because of stability issues. Both of these are honest or 'fair-use' actions for which the GenerationID may change. It is undesirable that the honest administrator should be punished by having the license server shut down for a lengthy period while he manually obtains a new license file from the producer. This is why Revenera recommends a nag message instead of a HostID failure. The advantage of the separate GID license is that a nag message can be removed by supplying (just) a new GID license - the served license is unaffected.

Compare this to trusted storage behavior. An accidental clone will cause TS to incur a binding break. However, a license administrator can recover quickly from a binding break by issuing an online repair or reinstall request to the producer's backoffice.

But if GenerationID is not used as the `SERVER` line HostID, what should be used? There are two options: `VM_UUID` or `ETHER`. Both of these change on a managed clone, but `ETHER` is generally easier to change back to its original value (via a VMM GUI) than `VM_UUID` is. This allows the honest system administrator to continue running his license server while talking to the producer about getting a new GID license in order to disable the nag message.

Revenera does not recommend using GenerationID for nag messages on license servers running on managed cloud platforms such as Amazon EC2. This is because GenerationID behavior cannot be guaranteed on such platforms. Instead, use `VM_UUID`, or `AMZN_EIP` on Amazon EC2.

Summary of Recommended Virtualization Functionality in FlexNet Publisher from 11.15.0

Deprecated functionality

In general, it is now difficult to recommend denying licensing operations in virtual environments, due to widespread uptake of virtualization in enterprises. Therefore the window of usefulness of keywords such as VM_PLATFORMS and PHY_* (in license files), VIRTUALIZATION_POLICY (in ASRs), and the `ls_allow_vm` vendor variable is closing.

VMW_* license-file keywords were introduced at a time when VMware was by far the predominant hypervisor, and applied only to VMware guests. Now that multiple hypervisor options are available, Revenera instead recommends hypervisor-generic solutions. For example, the VM_UUID and ETHER keywords are now recommended instead of VMW_UUID and VMW_ETHER respectively. From 11.12.1, TS binding behaves in such a hypervisor-generic manner.

Imbind and support of the related LMB_* license-file keywords have been removed from FNP.

FNP-Certificate

Server HostID

The recommended HostID on the SERVER line depends on the virtual environment.

Virtual Environment	Recommended SERVER line HostID	Comment
Hyper-V	ETHER	Hyper-V does not change the VM_UUID value on clone. For Windows guests, consider using a GenerationID nag message.
Parallels	ETHER	Parallels does not change the VM_UUID value on clone.
XenServer	VM_UUID or ETHER	For Windows guests, consider using a GenerationID nag message.
VMware	VM_UUID or ETHER	For Windows guests, consider using a GenerationID nag message.
Amazon EC2	AMZN_EIP or VM_UUID	Use Elastic IP for AMZN_EIP

Virtual Environment	Recommended SERVER line HostID	Comment
Oracle VirtualBox	VM_UUID	
QEMU-KVM		
Microsoft Azure		
Google Compute		
everRun	VM_UUID or ETHER	For Windows guests, consider using a GenerationID nag message.
Nutanix	VM_UUID or ETHER	

FPN does not support the VM_UUID keyword for clients. For node-locked certificate clients, ETHER or FLEXID (using USB passthrough) is the recommended HostID.

Similarly, in guests of hypervisors that are not currently detected by FPN, ETHER is the recommended HostID, since properly designed hypervisors will change MAC address on clone and retain it on migrate.

Client HostID

VM_UUID is not currently supported as a HostID on the INCREMENT line. However, one can call `lc_hostid(1m_job, VM_UUID, buf)` from the client, which means producers can use VM_UUID indirectly as a VDH (vendor defined hostid).

Alternatively, the ETHER HostID is the most flexible for clients on virtual environments.

Dongles and USB passthrough

One option in certificate models is to use the FLEXID HostID, with a dongle plugged into the USB port of the native host, and rely on USB passthrough support by the hypervisor to allow the FLEXID HostID to be extracted directly in the Guest. Publishers who consider this option should be aware of the following:

- There are two third-party layers outside of Revenera's control where bugs can manifest: the hypervisor and the dongle driver. These bugs can manifest on upgrade of the hypervisor, or the dongle driver, or both and are beyond Revenera's control for timely resolution.
- Such a configuration will prevent VM migration and is therefore not suitable for licensing in high-availability environments.
- A dongle with USB-passthrough will prevent cloning between hosts, but may not prevent cloning onto the same host. However, hypervisor providers may provide additional cloning-prevention functionality. For example, some models of FLEXID9-based dongles are accessible from only one VM instance on a VMware ESXi host (VMware KB 1021345).



Note - This extract-dongleID-in-just-one-VM feature is supported by VMware, not Revenera.

FNP-TS

Recommendations are:

- Upgrade to latest FNP and FNO, at least FNP 11.12.1+ and FNO 12.11+
- Set the FNO "Deny Repair if ACPI Generation ID is changed" policy to No. This allows maximum chance of recovery from fair-use revert-to-snapshot scenarios.
- Run TS-based license servers on Windows Server 2012+ guests to take advantage of GenerationID in binding.
- Recommendations for trials:
 - Avoid locally activated trials (ASRs) other than in trusted sites.
 - Instead of ASRs, consider allowing a backoffice-activated model for trials:
 - Provide a 'free' trial EntitlementID to customers
 - The free EntitlementID activates a feature-clipped and expiring license from the backoffice
 - Each enterprise customer gets their own trial EntitlementID with a limited count for the enterprise
 - Do not allow repair or reinstall against the free EntitlementID
- Because TS will break in a VM cloned for disaster recovery, it is recommended that disaster recovery centers be given a special 'recovery' EntitlementID that allows them to immediately establish a new license from a backoffice when a disaster recovery VM spins up.

Troubleshooting FNP-TS Behavior in Virtual Environments

Cloned VM's TS is trusted, Source VM's TS is untrusted

This can happen when exporting and importing a VM, or cloning a VM using Hyper-V: the imported/cloned VM is trusted, but the source VM from which the export/clone occurred is untrusted. From a licensing point of view, this is OK because at the end of the export/import/clone process there is still exactly one VM that has a trusted TS instance.

If a new VM is cloned from a hibernated source VM, then on first start the cloned VM's TS is not broken

This is a bug in the ACPI driver provided by Microsoft for extracting GenerationID - it does not initially pick up the new GenerationID provided by the hypervisor in this scenario. However, upon restart of the cloned VM, the new GenerationID is correctly extracted and the cloned VM's TS then incurs the expected binding break.

A Linux VM's TS does not break when cloned on Hyper-V

Hyper-V does not change VMID when a VM is cloned. Also, GenerationID is not supported in Linux guests. Hence we are dependent on the MAC address change to incur a binding break. In this case, the VM picks up the MAC address change (causing the TS break) only when the cloned VM is rebooted.

Cloned VM's TS is still trusted

Things to check:

- Is the source VM's TS still trusted? It may be that the TS in the source is now untrusted.
- Try restarting the license server.
- Try shutting down and restarting the cloned VM using the VMM interface.

Appendix: Generating a Vendor Daemon Nag Message

Process Flow for Generating and Using License Files

Step 1: The enterprise license administrator determines the GenerationID and ETHER values

Install the FlexNet Licensing Service first, for example with

```
installanchorservice <vendor name> <product name>
```

Then run

```
lmhostid -ptype VM -genid
```

Sample output is:

The FlexNet host ID of this machine is "VM_GENID=a5b88331fca36f5f:1ba819e285412a

Then run

```
lmhostid -ether
```

Sample output is:

The FlexNet host ID of this machine is "00155d0a9018"

The enterprise license administrator then provides both HostIDs to the producer.

Step 2: Producer generates license files

The producer generates two license files.

The first is the normal served license file, counted.lic say, with a SERVER line HostID of ETHER.

The second is a separate 'GID' license file, which looks like this:

```
INCREMENT gid demo 1.0 permanent uncounted \  
  VENDOR_STRING=VM_GENID= a5b88331fca36f5f:1ba819e285412a \  
  VENDOR_STRING=ETHER= 00155d0a9018
```

```
HOSTID=ANY SIGN= "..."
```

Step 3: License administrator runs license files with custom vendor dameon

Both GID.lic and counted.lic are sent to the enterprise license administrator.

Both license files are kept in the same directory.

The license server is started with counted.lic, with a vendor daemon customized as described below.

Adapting lsvendor.c to Generate a Nag Message when GenerationID Changes

An example lsvendor.c can be obtained from Revenera support, but the changes required are below.

```
/* Set up LOG macro - required when using lmadmin */
#ifndef LL_LOGTO_ASCII
#define LL_LOGTO_ASCII 1
#endif
#define LOG (x) {ls_log_prefix(LL_LOGTO_ASCII, 0); (void) ls_log_asc_printf x;}

/* ***** */
/* GenerationID nag message example - initialization */
/* ***** */
/* job handle used to perform GenerationID-related operations */
static LM_HANDLE* lm_genid_job = NULL;
/* vendor code structure used to perform GenerationID-related operations */
/*do not strip*/static VENDORCODE vcode_genid;

/** GID.lic is a separate uncounted license file - example is:
    INCREMENT gid demo 1.0 permanent uncounted \
        VENDOR_STRING=VM_GENID= a5b88331fca36f5f:1ba819e285412a \
        HOSTID=ANY SIGN= "... "
**/
/* Initialise lm_genid_job at vendor daemon startup, and load GID.lic */
void user_init1 ()
{
    int result = LM_NOERROR;

    /* Initialise the vcode structure and create lm_genid_job for later use*/
    /* we may need to call lc_new_job twice to succeed */
    result = lc_new_job(lm_job, lc_new_job_arg2, &vcode_genid, &lm_genid_job);
    if (LM_NOERROR != result)
    {
        result = lc_new_job(lm_job, lc_new_job_arg2, &vcode_genid, &lm_genid_job);
    }
    if (LM_NOERROR == result)
    {
        /* Ensure that lm_genid_job only uses GID.lic */
        (void) lc_set_attr(lm_genid_job, LM_A_DISABLE_ENV, (LM_A_VAL_TYPE)1);
        (void) lc_set_attr(lm_genid_job, LM_A_LICENSE_FILE, (LM_A_VAL_TYPE) "");
        /* using a separate job to load GID.lic means the license server will not serve the
           dummy 'gid' feature */
    }
}
```

```

    (void) lc_set_attr(lm_genid_job, LM_A_LICENSE_FILE, (LM_A_VAL_TYPE)"GID.lic");
}
else
{
    printf("Failed to create new job for GenerationID check. Error was %d\n", result);
    return;
}
/* initialise VM information by calling lc_virtualstatusget */
/* this works round a virtualization attributes initialization bug */
lc_virtualstatusget(lm_genid_job);
}
void (*ls_user_init1)() = user_init1;
/* release lm_genid_job resources on shutdown */
void vd_shutdown ()
{
    if (lm_genid_job)
    {
        (void) lm_free_job (lm_genid_job);
        lm_genid_job = NULL;
    }
}
void (*ls_vd_shutdown)() = vd_shutdown;

/* *****
 * This callback is called once a minute, and
 ** assumes lm_genid_job has already been initialized in one of ls_user_init1/2/3
 ** looks for a "gid" feature accesible via the lm_genid_job handle,
 ** then compares the runtime-extracted VM_GENID with that extracted from the "gid" feature,
 ** and prints a nag message to stdout if they are not the same.
***** */
void check_genid_periodic()
{
    char bufGenID[MAX_CONFIG_LINE]= ""; char bufTemp[MAX_CONFIG_LINE]= "";
    int result = LM_NOERROR;
    CONFIG* conf = NULL;

    time_t now = time(0);
    struct tm* ptm = localtime( &now );
    sprintf (bufTemp, "%02d:%02d:%02d WARNING:", ptm->tm_hour, ptm->tm_min, ptm->tm_sec);

    if (NULL == lm_genid_job)
    {
        LOG((" %s Cannot do GenerationID check - job creation failed\n", bufTemp));
        return;
    }
    conf = lc_get_config(lm_genid_job, "gid");
    /* check for the 'gid' feature and that it has a vendor string */
    if (NULL == conf || NULL == conf->lc_vendor_def)
    {
        LOG((" %s Generation ID is missing from license file\n", bufTemp));
        return;
    }
    /* Ensure the signature of the gid feature is valid */
    if (LM_NOERROR != lc_check_key(lm_genid_job, conf, &vcode_genid))
    {
        LOG((" %s Generation ID feature keycheck error: %s\n",

```

```

        bufTemp, lc_errstring(lm_genid_job)));
    return;
}
/* Extract runtime GenerationID value */
if (LM_NOERROR != lc_hostid(lm_genid_job, VM_GENID, bufGenID))
{
    /* this will indicate if the licensing service is not installed,
       or disabled, or the wrong version */
    LOG((" %s %s\n", bufTemp, lc_errstring(lm_genid_job)));
    return;
}
/* Compare runtime GenerationID to that from gid's vendor string */
if( strcmp( bufGenID, conf->lc_vendor_def) )
{
    LOG((" %s Generation ID should be %s, but is actually %s\n",
        bufTemp, conf->lc_vendor_def, bufGenID));
    return;
}
/* if we get here, all is good and no nag message is needed */
return;
}
void (*ls_daemon_periodic)() = check_genid_periodic;

```