Inventory Adapters and Connectors Reference

FlexNet Manager Suite 2019 R2

Legal Information

Document Name: FlexNet Manager Suite Inventory Adapters and Connectors Reference version 2019 R2 (for on-premises implementations)

Part Number: FMS-14.0.1-AR01

Product Release Date: December 16, 2019

Copyright Notice

Copyright © 2019 Flexera.

This publication contains proprietary and confidential technology, information and creative works owned by Flexera 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 is strictly prohibited. Except where expressly provided by Flexera in writing, possession of this publication shall not be construed to confer any license or rights under any Flexera intellectual property rights, whether by estoppel, implication, or otherwise.

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

FlexNet Manager Suite incorporates software developed by others and redistributed according to license agreements. Copyright notices and licenses for this externally-developed software are provided in the link below.

Intellectual Property

For a list of trademarks and patents that are owned by Flexera, see http://www.flexera.com/intellectual-property. All other brand and product names mentioned in Flexera 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.

Inventory Adapters and Connectors

FlexNet Manager Suite relies on software and hardware inventory collected from the computers in your computing estate to calculate what licenses are required. While the system includes a full inventory-gathering capacity, it also allows you to import inventory collected by other tools you may already have.

As well, the system uses a considerable amount of business-related data, including organizational structure, and records of purchases, to correctly track your existing license entitlements. This business data can also be imported from other sources in your enterprise. However, business-related data is not the subject of this document: instead, see *Using FlexNet Business Adapters*, which covers both the creation of business adapters and import of data through them.

This document, therefore, focuses on the collection of software and hardware inventory.

The data collected by the 'third party' tools usually needs to be rationalized in different ways, and mapped into the data fields within the FlexNet Manager Suite database. The interfaces that allow this mapping are called *adapters*, structured as XML files. Several adapters are provided by default with the system, or are available for download from the Flexera Product and License Center. You can also build custom adapters for inventory using the Inventory Adapter Studio, supplied with the product.

Some other inventory systems allow a rather simpler interface, where the preparation of XML-based adapters is not necessary. Instead, it is enough to declare a connection to these systems, most often from an inventory beacon. Some such systems are databases for third-party tools, where a direct connection may be possible, given a suitable connection string. The incoming data can then be mapped automatically into the operations databases for FlexNet Manager Suite, most often directly into the compliance database. Since the preparation of XML-based adapter files is not required in these cases, they are referred to simple as *connectors*.

This document gives priority to several of the standard adapters available for FlexNet Manager Suite, because these are the more complex cases that may require special installation, or custom XML files and the like. Over time, coverage of connectors will also increase.

Before diving into the details of the various connectors and adapters, some useful tables compare the kinds of information available from various data sources (see Inventory Comparison Matrix).

Following the details about available adapters and connectors, the Inventory Adapter Studio is documented. This is for advanced and experienced developers who wish to create custom adapters (see The Inventory Adapter Studio).

Between these 'bookends', the following adapters and connectors are documented, in this (alphabetical) order:

- ADDM — now rebranded as BMC Discovery (see below)
- App-V — see App-V Server Adapter
- Atrium — see BMC Atrium and Remedy Integration
- AWS EC2 — see AWS EC2 Connector
- BMC Discovery — see BMC Discovery (ADDM) Adapter
- Data Platform — see Data Platform Integration
- HP DDMI — see HP DDMI Adapter

- HPE Universal Discovery — see HPE Universal Discovery Adapter
- Microsoft Azure - see Microsoft Azure Connector
- Office 365 — see Managing Microsoft Office 365 Licenses
- Oracle Enterprise Manager — see Oracle Enterprise Manager Adapter
- Salesforce — see Salesforce Subscription Management
- ServiceNow — see ServiceNow Integration with FlexNet Manager Suite
- Tanium Connector — see Tanium Connector
- XenApp server adapter — see XenApp Server Adapter.

Contents

1. Inventory Comparison Matrix 16

Part I. BMC Discovery (ADDM) Adapter 20

1. Choosing a Configuration 21

How the Adapter Works 21

Components Explained 22

Optional Patterns 22

The FlexNet inventory agent 25

Database Table Creation 26

The Adapter Executable 26

2. Installation and Configuration 30

Choosing a Staging Server 30

Creating the Staging Database Tables 31

Installing and Configuring the Staging Tool 32

Installing the FlexNet inventory agent 33

Configuring BMC Discovery 34

Installing Optional Patterns 34

Enabling Optional Patterns 36

Rediscovering Affected Computers 36

Account Configuration 37

Validation and Operation 38

3. Known Issues 40

4. Appendix A: Details of Patterns 41

Overview of Patterns 41

FileEvidence 42

To configure the FileEvidence pattern 44

InstallAnywhereEvidence 44

InstallShieldMultiplatformEvidence 45

UnixHardwareData 46

WindowsLastLoggedOnUser 49

5. Appendix B: Data Mappings 51

Source to Staging **51**
ADDMVersion_ci (Staging Table) 52
Cluster_ci (Staging Table) 52
ClusterHost_ci (Staging Table) 53
CPUInformationDetail_ci (Staging Table) 54
DiscoveredPackages_ci (Staging Table) 55
DiscoveredService_ci (Staging Table) 56
DiscoveredVirtualMachine_ci (Staging Table) 57
FileEvidenceDetail_ci (Staging Table) 58
FileSystem_ci (Staging Table) 58
HardwareEvidenceDetail_ci (Staging Table) 59
Host_ci (Staging Table) 62
HostContainerProcessorInfo_ci (Staging Table) 64
HostDetail_ci (Staging Table) 65
HostInfo_ci (Staging Table) 65
InstallerEvidenceDetail_ci (Staging Table) 67
LastLoggedOnUserDetail_ci (Staging Table) 67
NetworkInterface_ci (Staging Table) 68
SoftwareInstance_ci (Staging Table) 69
SoftwareInstanceVirtualMachine_ci (Staging Table) 69
Staging to FlexNet Manager Suite **70**
Inventory Device (Computer) 71
Installer Evidence 75
File Evidence 76
WMI Evidence 78
Part II. App-V Server Adapter **80**
1. Architecture, Components, and Prerequisites **81**
Architecture and Operation for App-V 4.6 ..... **81**
Architecture and Operation for App-V 5 and Later **83**
2. Set Up and Operations **87**
Obtaining (and Deploying) the Adapter Components **87**
Command Line for PowerShell Script 90
File Format for .raa 92
Configuring the Adapter **94**
Import Evidence and Recognize Applications **99**

3. Issues and Limitations 103
Limitations 103
Investigating Issues 104
Known Issues 107
4. Data mapping 109
App-V Release 4.6 Data Transfers 109
App-V Release 5.0 (and Later) Data Transfers 110
Part III. BMC Atrium and Remedy Integration 113
1. Architecture, Operation, and Prerequisites 114
Architecture and Operations 114
Prerequisites 116
Obtaining the Adapter Components 117
2. Installing the Adapter 119
Preparing the Databases 119
Creating a View Only SQL User in FlexNet Manager Suite 122
Reserving an Atrium Dataset 123
Importing Atrium Jobs and Transforms 124
Configure Atrium Jobs 125
Install the Business Importer and Link with Remedy 128
Tuning Creation of Configuration Items 135
Scheduling the Adapter 136
Verifying Data Export to BMC Atrium 138
3. Additional Customization of the Adapter 140
Preparing for Large Datasets 140
Updating Connection Details 141
Updating Email Details 143
Turning Off Email Alerts for Export Errors 145
Changing the Log File 145
4. Appendices 147
Appendix 1: Export of Computers 147
Appendix 2: Export of Applications/Products 153
Appendix 3: Import of Assets 157
Part IV. AWS EC2 Connector 161
1. Prerequisites and Setting Up 163

2. Thinking about Inventory and Licensing 165
Collecting Inventory from Instances 165
BYOSL Licensing Considerations 169
3. Images for Different Kinds of Instances 172
Configuring an AMI for Short-Lived Instances 172
Configuring an AMI for Longer-Life Instances 179
4. Appendices: Technical Data 185
Appendix 1: Cmdlets in the AWS EC2 connector 185
Appendix 2: Data Imported by AWS EC2 connector 186
Appendix 3: Enhanced Inventory Gathered by Agent 188
Part V. Microsoft Azure Connector 189
1. Prerequisites and Setting Up 191
2. Thinking about Inventory and Licensing 192
Collecting Inventory from Instances 192
BYOSL Licensing Considerations 196
3. Configuring an Azure VM Image for Instances 199
4. Appendices: Technical Data 205
Appendix 1: Cmdlets in Azure PowerShell 205
Appendix 2: Data Imported by Microsoft Azure Connector 205
Appendix 3: Enhanced Inventory Gathered by Agent 206
Part VI. Data Platform Integration 208
1. Integration with Data Platform v5 209
2. Configuring the Data Platform Connector 213
3. Data Mappings, Gaps, and Impacts 216
Comparison of Sources 217
Imported Installer Evidence 219
Imported Computers (Inventory Devices) 221
Imported VMs and Hosts 230
Imported Installation Records 233
Imported Users 234
Imported Software Usage 236
Part VII. HP DDMI Adapter 238

1. Purpose and Architecture of the HP DDMI Adapter 239
2. Installation and Configuration 241
Download Adapter Tools Archive 241
Creating the Staging Database 242
Configuring the FlexNet Agent for HP DDMI 245
Configuring Upload and Import Connection 247
Part VIII. HPE Universal Discovery Adapter 249
1. Selecting a Configuration 250
Architecture and Working of the HPE Universal Discovery Adapter 250
The Adapter Executable 251
2. Installation and Configuration 253
Download Adapter Tools Archive 253
Selecting a Staging Server 254
Creating the Staging Database Tables 254
Configuring HPE Universal Discovery System 255
Installing and Configuring the Staging Tool 256
3. Operation and Validation 258
HPE-UD Adapter Operation 258
Validating the HPE-UD Adapter 258
Part IX. Managing Microsoft Office 365 Licenses 260
1. Microsoft Office 365 License Management Considerations 261
2. Managing Office 365 Licenses through FlexNet Manager Suite 264
3. Connecting to Microsoft Office 365 266
Prerequisites and Configuration Considerations 267
Creating Connections using the Microsoft 365 Connector 269
Using FlexNet Manager Suite's Multi-Tenant App to Connect to Microsoft 365 269
Registering an app using the Azure portal to connect to Microsoft 365 271
Configuring Token Lifetimes in Azure Active Directory 273
Creating Connections using the Microsoft Office 365 (Deprecated) Connector 275
Troubleshooting Imports from Office 365 277
Troubleshooting Microsoft 365 Connector Imports from Office 365 278
Troubleshooting Microsoft Office 365 (Deprecated) Connector Imports from Office 365 279
4. Migrating to a New Microsoft Connector 282

Part X. Oracle Enterprise Manager Adapter 283

1. Understanding the Oracle Enterprise Manager Adapter 284

How the Adapter Assists in Inventory Gathering 284

Prerequisites for the OEM Adapter 286

Components 286

Download Adapter Tools Archive 287

2. Installing the Adapter, and More 288

Installing the OEM adapter 288

Grant Permissions to Account 292

Other Setup Activities 293

Inventory-Gathering Accounts on Oracle Servers 294

Save Inventory Account in Password Manager 297

Assign Beacon to Subnet 297

Configure Collection of Oracle Inventory 298

3. Modifying the Adapter 303

Reconfiguring the OEM Adapter 303

Updating Connection Details 303

Configure Data Staging 305

Managing Email Alerts 306

Configure Logging 307

Part XI. Salesforce Subscription Management 308

1. Salesforce License Considerations 309

2. Viewing Salesforce License Information with FlexNet Manager Suite 310

3. Connecting to the Salesforce Online Service 312

Managing Connections to Salesforce.com 312

Part XII. ServiceNow Integration with FlexNet Manager Suite 317

1. Key Concepts 319

Data Types that Can Be Merged 319

How Data Is Merged 320

Source of Truth 320

Where to View Merged Data 320

ServiceNow Computer and Application Records 321

2. Architecture, Components, and Prerequisites 324

Architecture **324**

Prerequisites **325**

Download Adapter Tools Archive **327**

3. Installation and Configuration **328**

Setting Up the Integration **328**

Installing the FlexNet Manager Suite Integration Application from the ServiceNow Store 328

Creating a ServiceNow Integration User 329

Setting Up Data Flows from FlexNet Manager Suite to ServiceNow **330**

Configuring ServiceNow for Import 331

Configuring FlexNet Manager Suite for Export 331

Configuring the Utility for Export from FlexNet Manager Suite 333

Setting Up Data Flows from ServiceNow to FlexNet Manager Suite **335**

Setting Up a MID Server 336

Configuring ServiceNow for Export 337

Configuring FlexNet Beacon for Import 338

4. Operational Details **341**

Process for Exports from FlexNet Manager Suite to ServiceNow **341**

Import Runs Columns 343

Import Transactions Columns 344

Transform Maps for ServiceNow Integration 345

Command-Line Tool for Export to ServiceNow 351

Process for Exports from ServiceNow to FlexNet Manager Suite **355**

Properties for Export Columns 356

Business Adapter Mappings 357

5. Appendices **360**

Integration Properties **360**

Additional ServiceNow Indexes for Performance **362**

Removing a Legacy Integration Application **362**

Configuring the Software Asset Management Foundation Plugin **363**

Deleting Records from ServiceNow **365**

Part XIII. Tanium Connector **366**

1. Prerequisites **367**

FlexNet Beacon Server Prerequisites **367**

Tanium Server Prerequisites **367**

Creating a Flexera SQL Edition view within Tanium Asset 368
Installing Tanium Connector on FlexNet Manager Suite 369
2. Configure FlexNet Beacon to connect with Tanium 371
3. Validating Imports to the Inventory Beacon 373
4. Tanium Connector Architecture 375
5. Investigating Tanium Connection Errors 377
6. Appendix: Technical Data 379
Part XIV. XenApp Server Adapter 382
1. Architecture, Operations and Prerequisites 384
Architecture and Operation 384
Prerequisites 392
2. Setting Up the XenApp Server Adapter 394
Creating the Staging Database 394
Installing the XenApp Server Agent 396
Create a Scheduled Task 397
Create Connections for Data Upload 400
3. Command-Line Options 405
XenApp Server Agent Command Line Options 405
4. Validation, Troubleshooting, and Limitations 410
Validation and Problem Solving 410
Limitations 411
5. Database Impacts 413
Affected Database Tables 413
Part XV. The Inventory Adapter Studio 415
1. What Is Inventory Adapter Studio? 416
2. Cautions, Prerequisites, and References 417
3. The Inventory Adapter Studio Interface 419
Toolbar 420
Step Explorer 422
Edit panel 423
4. Installing Inventory Adapter Studio 429
5. Understanding Inventory Adapters 430

The Architecture of Compliance Importer 430
Structure of an Inventory Adapter 431
Structure of Templates for Inventory Adapters 435
6. Creating a New Adapter 436
7. Editing an Existing Adapter or Template 438
8. To Create a Source Connection 440
9. Overview: Process for Developing an Inventory Adapter 443
Adding a New Step to an Inventory Adapter 444
Removing a Step from an Inventory Adapter 444
Reordering Steps in an Inventory Adapter 445
10. Disconnected Mode 446
Selecting a Step for Connected or Disconnected Modes 446
Why Special Steps Are Required for Disconnected Mode 447
11. Tips for Editing an Adapter 449
12. To Save an Adapter 453
13. Testing an Adapter 454
To Run a Full Import 454
To Diagnose Readers for Your Adapter 455
Diagnosing Writers for Your Adapter 455
14. Publishing Your Adapter 458
15. Inventory Adapter Object Model 460
Inventory Object: AccessingDevice 460
Inventory Object: AccessingUser 461
Inventory Object: ActiveDirectoryComputer 461
Inventory Object: ActiveDirectoryDomain 462
Inventory Object: ActiveDirectoryExternalMember 462
Inventory Object: ActiveDirectoryGroup 463
Inventory Object: ActiveDirectoryMember 463
Inventory Object: ActiveDirectoryUser 464
Inventory Object: ActiveSyncDevice 464
Inventory Object: ClientAccessEvidence 466
Inventory Object: ClientAccessEvidenceMapping 466
Inventory Object: ClientAccessedAccessEvidence 467

Inventory Object: ClientAccessedAccessOccurrence 468
Inventory Object: Cluster 469
Inventory Object: ClusterGroup 470
Inventory Object: ClusterGroupMember 471
Inventory Object: ClusterHostAffinityRule 471
Inventory Object: ClusterNode 472
Inventory Object: Computer 473
Inventory Object: ComputerCustomProperty 478
Inventory Object: ConsolidatedAccessEvidence 478
Inventory Object: ConsolidatedCluster 481
Inventory Object: ConsolidatedClusterGroup 482
Inventory Object: ConsolidatedClusterHostAffinityRule 483
Inventory Object: ConsolidatedComputer 484
Inventory Object: ConsolidatedFileEvidence 491
Inventory Object: ConsolidatedInstallerEvidence 495
Inventory Object: ConsolidatedOracleDatabaseUser 499
Inventory Object: ConsolidatedRemoteAccessFile 501
Inventory Object: ConsolidatedRemoteAccessInstaller 504
Inventory Object: ConsolidatedVMPool 505
Inventory Object: ConsolidatedWMIEvidence 505
Inventory Object: Domain 507
Inventory Object: EvidenceAttribute 508
Inventory Object: FileEvidence 509
Inventory Object: ILMTPVUCounts 510
Inventory Object: InstalledFileEvidence 511
Inventory Object: InstalledFileEvidenceUsage 512
Inventory Object: InstalledInstallerEvidence 514
Inventory Object: InstalledInstallerEvidenceAttribute 515
Inventory Object: InstalledInstallerEvidenceUsage 517
Inventory Object: InstalledWMIEvidence 519
Inventory Object: InstallerEvidence 520
Inventory Object: InstallerEvidenceRepackageMapping 521
Inventory Object: Instance 522
Inventory Object: InstanceUser 524
Inventory Object: LicenseUser 525

Inventory Object: RelatedInstalledInstallerEvidence 526
Inventory Object: RemoteUserToApplicationAccess 528
Inventory Object: Site 530
Inventory Object: SiteSubnet 530
Inventory Object: SoftwareLicense 531
Inventory Object: SoftwareLicenseAllocation 531
Inventory Object: User 533
Inventory Object: VDI 534
Inventory Object: VDITemplate 536
Inventory Object: VDIUser 538
Inventory Object: VMHostManagedBySoftware 538
Inventory Object: VMPool 539
Inventory Object: VirtualMachine 540
Inventory Object: WMIEvidence 543

1

Inventory Comparison Matrix

Not all inventory sources are created equal. Some sources do better at collecting details necessary for licensing calculations than others. The following tables let you quickly assess what data may be missing from various sources.

Strictly speaking, not all the products listed in these tables are inventory sources (particularly in the later tables). However, they are all *data* sources that can provide data for FlexNet Manager Suite to use in license compliance calculations.

Note: *For supported versions of these products, please review the Release Notes for the current version of FlexNet Manager Suite.*

Inventory Sources

Information Collected per Product									
Computer	User	Product code	Evidence		Usage	Virtualization		Clusters	Oracle DB
			Installer	File		Server	Application		
FlexNet inventory agent (previously Managesoft)									
Yes	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Microsoft SCCM (previously SMS)[1]									
Yes	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	No	**Yes**	No	No
BDNA Normalize (Data Platform)[2]									
Yes	**Yes**	No	No	No	No	No	No	No	No
BMC Discovery (previously ADDM)									
Yes	**Yes**	**Yes**	No	**Yes**	No	**Yes**	No	**Yes**	No
BMC BladeLogic Client Automation (previously Marimba)[3]									
Yes	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	*Partial*	No	No	No
HP Discovery and Dependency Mapping Inventory (DDMI)									

Information Collected per Product									
Computer	User	Product code	Evidence		Usage	Virtualization		Clusters	Oracle DB
			Installer	File		Server	Application		
Yes	**Yes**	**Yes**	**Yes**	**Yes**	No	No	No	No	No
HPE Universal Discovery (HP-UD)									
Yes	**Yes**	**Yes**	**Yes**	**Yes**	No	No	No	No	No
IBM License Metric Tool (ILMT) on Microsoft SQL Server									
Yes	No	**Yes**	No	No	No	**Yes**	No	**Yes**	No
IBM License Metric Tool (ILMT) on IBM DB2									
Yes	No	**Yes**	No	No	No	**Yes**	No	**Yes**	No
BigFix platform (previously Tivoli Endpoint Manager and IBM BigFix)[4]									
Yes	**Yes**	**Yes**	No	No	No	No	No	No	No
Microsoft Exchange ActiveSync[5]									
Yes	**Yes**	No	No	No	No	No	No	No	No
Symantec IT Management Suite (formerly Altiris)[6]									
Yes	**Yes**	**Yes**	No	**Yes**	No	*Partial*	No	No	No

1. App-V integration provides application virtualization in Microsoft SCCM (previously SMS).

2. Doesn't support BDNA running on Oracle DB. v5 was released April 9th, 2018 as a separate download.

3. Virtualization: VMware, and Solaris Zones. Collects VM properties, but doesn't collect host-VM relationships.

4. Doesn't collect publisher information, and has potential for false recognition positives.

5. Mobile devices only.

6. Virtualization: VMware, and Hyper-V. Doesn't collect pool or resource allocations.

Server Virtualization

Information Collected per Product									
Computer	User	Product code	Evidence		Usage	Virtualization		Clusters	Oracle DB
			Installer	File		Server	Application		
*Microsoft Hyper-V**									
						Yes		**Yes**	
Oracle VM Server for x86									
						Yes		**Yes**	
*VMware vCenter**									

Information Collected per Product									
Computer	User	Product code	Evidence		Usage	Virtualization		Clusters	Oracle DB
			Installer	File		Server	Application		
						Yes		**Yes**	
*VMware ESXi Server**									
						Yes			
*VMware vSphere**									
						Yes			

* Refer to the **Inventory Sources** table above for sources that provide data for these technologies.

Application Virtualization

Information Collected per Product									
Computer	User	Product code	Evidence		Usage	Virtualization		Clusters	Oracle DB
			Installer	File		Server	Application		
App-V Server[1]									
Partial	*Partial*	**Yes**	No	No	**Yes**	No	**Yes**	No	No
App-V integrated with SCCM[2]									
Yes	**Yes**	**Yes**	No	No	**Yes**	No	**Yes**	No	No
EdgeSight for Citrix Virtual Apps (previously XenApp EdgeSight)[3]									
Yes	**Yes**	*Partial*	No	**Yes**	**Yes**	No	**Yes**	No	No
Citrix XenApp server									
Yes	**Yes**	**Yes**	No	**Yes**	**Yes**	No	**Yes**	No	No
Citrix Virtual Desktops[4]									
							Yes		

1. Requires mapping of packages to applications within FlexNet Manager Suite. Requires Active Directory integration and FlexNet inventory agent.

2. Provided by SCCM adapter.

3. Installer evidence is gathered when combined with Flexera's XenApp server agent.

4. Provided by a dedicated FlexNet inventory agent.

Other System Integrations

Information Collected per Product									
Computer	User	Product code	Evidence		Usage	Virtualization		Clusters	Oracle DB
			Installer	File		Server	Application		
*Oracle Database**									
									Yes
*Oracle E-Business Suite**									
									Yes
Oracle Enterprise Manager[1]									
Salesforce									
	Yes								
Office 365									
	Yes								
*SAP BASIS**									
	Yes								
Tanium Asset									
Yes	**Yes**		**Yes**	**Yes**					

* Provided by a dedicated FlexNet inventory agent.

1. Provides Oracle database connection details for use with a dedicated FlexNet inventory agent.

ITSM System Integrations

Information Collected per Product									
Computer	User	Product code	Evidence		Usage	Virtualization		Clusters	Oracle DB
			Installer	File		Server	Application		
ServiceNow[1]									
BMC Atrium CMDB and Remedy[2]									

1. Export computers and installed applications to ServiceNow. Import contracts and assets from ServiceNow into FlexNet Manager Suite.

2. Export computers and installed applications to Atrium. Import asset details from Remedy and Atrium into FlexNet Manager Suite.

BMC Discovery (ADDM) Adapter

The tool from BMC for collecting hardware and software information, previously known as Atrium Discovery and Dependency Mapping (ADDM), from version 11 has been renamed BMC Discovery. This tool can be a useful inventory source as input to FlexNet Manager Suite to help in calculating license consumption as part of assessing your overall license compliance.

To collect inventory information from BMC Discovery and import into the operations databases maintained by FlexNet Manager Suite requires a data adapter. The adapter is documented in the following chapters.

Supported versions

The BMC Discovery adapter supports inventory import from the following releases of the BMC tool: 9.0, 10.0–10.2 (ADDM), 11.0–11.5

1

Choosing a Configuration

The adapter to extract data from BMC Discovery can operate at different levels of detail, and with different overheads, that depend on what level of licensing information you need to collect. This section gives a brief overview of how the adapter works, and explains the different configurations and how to choose between them. You need to choose the configuration appropriate for your enterprise before implementing the adapter.

How the Adapter Works

Although it is downloaded as a single zipped archive, the adapter includes several components to improve your license reporting. The overview of the finished system shows:

- An optional set of patterns that can be deployed to each BMC Discovery instance to improve the initial level of inventory detail collected by BMC Discovery.
- A staging server, which includes a simple SQL database where data collected from BMC Discovery can be massaged for upload to FlexNet Manager Suite. The executable that speeds the data extraction process also resides here. For on-premises solutions, this staging database may optionally reside on the central operations databases server.
- The conversion and upload component, which converts data formats and uploads the result to the central operations databases maintained by FlexNet Manager Suite.

Operation is summarized in this high-level diagram:

Figure 1: Process overview

The diagram shows:

1. Optionally making use of additional patterns packaged with the adapter download, inventory details are collected by BMC Discovery (previously ADDM).
2. The executable `FnmpADDMStage.exe` uses the web API to extract the information from BMC Discovery. This method is used because the export process is 100 times faster than using mappings, and there is no requirement to configure custom mappings for each BMC Discovery instance. This executable optionally writes the gathered data into local XML files (available for inspection when required), and (determined by the mode of operation) also writes the data from these XML files into the staging database.
3. A standard component of FlexNet Manager Suite called a Compliance Reader, executing from the central application server, collects data from the staging database and imports it into the operations databases. This final stage occurs when you run an inventory import to process the incoming inventory.

Components Explained

This section describes each of the components in more detail, with information to help you decide which of these you will implement in your enterprise. Installation and setup follows in a separate section. The components described below are:

- Optional collection patterns to enrich the inventory detail collected by BMC Discovery (see Optional Patterns)
- The FlexNet inventory agent (see The FlexNet inventory agent)
- The scripts for creating the staging database in Microsoft SQL Server (see Database Table Creation)
- The `FnmpADDMStage.exe` executable to extract inventory information from BMC Discovery, optionally write it to XML files, and insert it into the staging database (see The Adapter Executable).

Optional Patterns

There are six collection patterns in all.

- `FileEvidence` allows collection of file evidence for use within FlexNet Manager Suite. This evidence may be of two sub-types:
  - Executable files that form part of the application, which are gathered only for Window platforms, and may allow application recognition (particularly to the level of editions and versions)
  - Identification files including ISO tag files, which may be gathered across any platform (including Windows and UNIX-based environments).

The next two collection patterns run both on Windows and on non-Windows devices, instructing the BMC Discovery inventory agent to gather additional data:

- `InstallAnywhereEvidence` returns the list of package titles found in the repository maintained by Flexera's InstallAnywhere
- `InstallShieldMultiplatformEvidence` returns the list of packages installed by InstallShield Multiplatform, an earlier installation technology developed by Flexera.

The next collection pattern is for non-Windows devices only, and uses the FlexNet inventory agent to capture

additional data elements and integrate these with the BMC Discovery inventory collected in the standard way:

- `UnixHardwareData` gets accurate hardware details to allow license metrics for capacity-based license calculations (such as for Processor, Core, IBM PVU, and other license types).

Note: *This pattern relies on the installation of two files, `ndtrack.sh` and `ndtrack.ini`, that are not included with the adapter package. These files are available with any installation of the FlexNet Beacon or application server. For more information, see The FlexNet inventory agent.*

The last collection pattern contains instructions for the BMC Discovery inventory agent to pull additional data from the Windows Registry and WMI on Windows-based computers:

- `WindowsLastLoggedOnUser` recovers information about end-users that is required for user-based licensing (such as Named User license types).

Note: *There is no equivalent data available for UNIX-like systems.*

To help you assess which of the patterns you wish to use, the following table summarizes the available collection patterns. The default state, whether the pattern is enabled or disabled, is shown in the **Pattern** column. The **Footprint** column details the additional installation impact (other than loading the pattern into BMC Discovery) required for the collection pattern (the downside of using the pattern), and the **Impact** column shows what will *`not`* work if you omit this pattern (the downside of *`not`* using it).

Pattern	Footprint	Impact of omitting pattern
`FileEvidence` Default: Functionality is controlled in two parts: • Gathering identity (or tag) files (all platforms) is enabled by default • Gathering executable evidence (Windows only) is disabled by default.	No installation required. This pattern is configurable for paths searched (per platform), and for file name extensions used for tag files. Search times on target machines will depend on configuration and the numbers of installed applications found.	An application that • Is not already correctly recognized by BMC Discovery (perhaps because it is installed but not currently running), and • Relies exclusively on file evidence (as distinct from installer evidence) for recognition within FlexNet Manager Suite will not be recognized for inventory collected through BMC Discovery without this pattern. While the Application Recognition Library rarely makes use of file evidence for Windows-based applications, some publishers including IBM and Oracle make use of special identity files. Software identity (SWID) tags are also in increasing use, and these are identified using this pattern. Likely impact of omitting this pattern in total is medium (through the loss of identity files). Likely impact of leaving executable gathering turned off is low. ***Note:*** *Within FlexNet Manager Suite, file evidence is also required for application usage tracking; but no application usage tracking is possible through the BMC Discovery inventory tool.*
`InstallAnywhere Evidence` Default: Enabled	No installation required.	BMC Discovery does not recognize installation evidence from InstallAnywhere. Unless it recognizes the application by other means, the application will be missed without this pattern.
`InstallShield MultiplatformEvidence` Default: Enabled	No installation required.	BMC Discovery does not recognize installation evidence from InstallShield Multiplatform. Unless it recognizes the application by other means, the application will be missed without this pattern.

Pattern	Footprint	Impact of omitting pattern
`UnixHardwareData` Default: Enabled	Requires less than 13 MB installation (`ndtrack.sh` and `ndtrack.ini`), either on the hard disk of the target machines, or on a network share accessible to them all. Run-time is a second or so when triggered by BMC Discovery.	Without this, BMC Discovery does not capture sufficient hardware attributes from servers to support license consumption calculations on license types based on hardware capacity metrics (such as Processor, Core, IBM PVU, and other license types). Assess impact based on the license types you need to support through BMC Discovery inventory (mandatory for capacity metrics).
`WindowsLast LoggedOnUser` Default: Enabled	No installation required.	Without this pattern, BMC Discovery does not report any end-user identification that is needed for licenses requiring identification of the individual (such as Named User license types). Note that for general user-based licensing, in the absence of end-user identities, FlexNet Manager Suite will calculate every installation of such software as usage by an unknown end-user, so that only license types depending specifically on identity will be affected. If you wish to allocate license entitlements to specific individuals, you also require this identification. Impact: medium.

Note: *For releases of BMC Atrium Discovery and Dependency Mapping up to and including 11.0, there is an additional pattern that improves the details about processors collected by BMC Discovery. This level of detail is required only for Oracle and IBM license calculations (for other licenses, the patterns above are adequate). By contractual arrangement with BMC, this pattern can be provided only to customers approved by BMC. For further information, ask your Flexera consultant, who can arrange the necessary approval and provide the pattern for you. Notice that for BMC Discovery release 11.1 or later, the additional pattern is deprecated. BMC Discovery 11.1 (and later) has improved processor information collection which makes this additional pattern unnecessary.*

For more information:

- About each of these patterns, please see Appendix A: Details of Patterns
- About installing these patterns, see Installing Optional Patterns
- About enabling or disabling each of the patterns, see Enabling Optional Patterns.

The FlexNet inventory agent

The FlexNet inventory agent is a standard component of any installation of FlexNet Beacon, and of the application server that is included with FlexNet Manager Suite. For this reason, the necessary files are not included in the BMC Discovery adapter zip archive, since they are already present in any standard product implementation. Installation to suit the BMC Discovery adapter is described in Installing the FlexNet inventory agent.

For use on Linux or UNIX platforms, the agent has two component files:

- `ndtrack.sh`, an agent responsible for collecting inventory details (in this case, from the BMC Discovery inventory system) and writing them in an intermediate format to a data file, ready for upload to an application server. The script has no active elements until it is triggered by BMC Discovery through one of the enhanced collection patterns.
- `ndtrack.ini`, a text file that contains configuration variables for `ndtrack.sh`.

The combined disk space requirement of both files is under 13MB.

Database Table Creation

The staging server requires an operating version of Microsoft SQL Server 2012 or later. Any edition is suitable, including Microsoft SQL Server Express (if its limitations on CPU, RAM, and database size are adequate for your purposes). As described in Choosing a Staging Server, the staging database may be located on a separate staging server, or on your central (on-premises) operations databases on your FlexNet Manager Suite server.

In the `SQL\` subdirectory of your unzipped adapter archive, the script `ADDM_staging.sql` is provided for creating the staging database, its tables, and its stored procedure. Obviously, this must be run on the SQL Server instance that is to host the staging database.

Important: *If you have been using an earlier version of ADDM, and are now migrating to BMC Discovery release 11 (or later), you must run the same `ADDM_staging.sql` script to update your staging database schema. If you are continuing to use ADDM release 10 (or earlier), the update to the staging database schema is not required, and you may omit it for now; but if you choose to apply the upgrade now (so that the staging database is prepared for any future migration to BMC Discovery 11 or later), you must also use the latest version of the `FnmpADDMStage.exe` executable from the same `Adapter Tools for` FlexNet Manager Suite 2019 R2`.zip` archive. This executable transfers to BMC Discovery data (for any BMC Discovery/ADDM version) to the staging database, and is schema-aware for the upgraded staging database.*

The Adapter Executable

BMC Discovery supports two ways of extracting inventory data it has collected:

- Using export mapping sets
- Using a web API.

While the former is more commonly used, the BMC Discovery adapter for FlexNet Manager Suite uses the latter, for the following reasons:

- It avoids the need to deploy and maintain export mapping sets on every BMC Discovery instance in your enterprise.
- Performance of data collection can be 100 times faster using the web API. For example, data extraction that can take over 24 hours using mapping sets can be completed in 20 minutes with the web API. This is because the export feature in BMC Discovery is designed for more complex export capabilities and can therefore be slow to perform simple queries. FlexNet Manager Suite requires only simple queries to be executed on BMC Discovery, and these are performed far more efficiently using the web API.

The tool to query the web API consists of two parts:

- `FnmpADDMStage.exe` — A .NET 4.5 console program capable of querying the XML API of BMC Discovery and writing the results into an SQL Server database, and optionally to XML files on the local file system. This program supports command line arguments, available using `FnmpADDMStage -h`
- `FnmpADDMSettings.xml` — The self-documenting configuration file for `FnmpADDMStage.exe` which contains the queries executed against BMC Discovery (in the BMC Discovery query language), and can include connection settings for BMC Discovery and SQL Server.

In operation, the executable, `FnmpADDMStage.exe`, extracts the inventory data from BMC Discovery and saves it for further processing. There are different ways that it can save the data, based on the following values of its `method` parameter:

- `Stage` — Summary: **BMC Discovery to XML**. Inventory gathered from BMC Discovery is saved to a series of XML files on the staging server. It is not imported into the staging database. The XML files allow for review of the gathered data, but the inventory is not imported into FlexNet Manager Suite from these files.

  Tip: *The XML file option also allows for disconnected scenarios, where inventory collected from an BMC Discovery server that is out of reach of the staging server can be written to XML, manually copied and transferred to another staging server, and the upload process resumed. See also the `Prestaged` method below.*

- `Staged` — Summary: **BMC Discovery to XML/SQL**. Inventory gathered from BMC Discovery is first written to the XML files on disk (for example for review), and then copied into the staging database where it can be imported into FlexNet Manager Suite for use in compliance calculations.
- `Prestaged` — Summary: **XML to SQL**. In this mode, inventory is not gathered from BMC Discovery. Instead, the XML files present on the disk from a previous inventory collection (and perhaps reviewed and approved by a human agent in this format) are now copied into the staging database where it can be imported into FlexNet Manager Suite for use in compliance calculations.
- `Stream` — Summary: **BMC Discovery to SQL**. Inventory is gathered from BMC Discovery, and loaded into the staging database where it can be imported into FlexNet Manager Suite for use in compliance calculations. In this method, inventory is not recorded in XML files on the staging server.

Default values for the `method` and all other parameters are set in the companion `FnmpADDMSettings.xml` file, and these are the values used when the executable is triggered (or run) without other command-line options. The settings file is self-documented, and the matching command-line options are available using `FnmpADDMStage -h`

When the executable writes XML files to (or reads them from) the local disk on the staging server, the files include the following (details are available in `FnmpADDMSettings.xml`):

Filename	Content
`Cluster.xml`	Details for each cluster.
`ClusterHost.xml`	Computers (host nodes) that are members of a cluster, including a key to the Cluster node to identify that cluster.
`CPUInformationDetail.xml`	Details of computer processors.
`DiscoveredPackages.xml`	Raw installer evidence gathered by BMC Discovery from the installer technologies supported by each operating system.

Filename	Content
`DiscoveredService.xml`	A report of particular services, used to help identify capabilities of hosts. This includes the VMMS service used to identify Windows machines with the Hyper-V role enabled.
`DiscoveredVirtualMachine.xml`	Raw results from BMC Discovery querying the list of virtual machines on a virtualization host.
`FileEvidenceDetail.xml`	File evidence produced by the `Flexera.FNMP.InventoryRawData.FileEvidence` pattern, covering software tag files and Windows executables.
`FileSystem.xml`	Name and size of all local file systems, used to approximate the total disks and storage of host.
`HardwareEvidenceDetail.xml`	Hardware details produced by the `Flexera.FNMP.InventoryRawData.UnixHardwareEvidence` pattern, using information gathered by the FlexNet inventory agent.
`Host.xml`	Details of all hosts known by BMC Discovery including their host name, operating system details, unique identification, processor, memory, and other hardware details.
`HostInfo.xml`	Raw host details not represented in a Host node, mainly the raw LPAR information from an IBM AIX LPAR environment.
`HostContainerProcessorInfo.xml`	Processor details for systems that are a container for hosts. For example, a container is a device partitioned into a number of logical hosts.
`InstallerEvidenceDetail.xml`	Installation evidence gathered using the patterns in the BMC Discovery adaptor, including evidence from installations by Install Anywhere and InstallShield Multi-platform.
`LastLoggedOnUserDetail.xml`	Details of the last logged-on user for Windows systems. (There is no equivalent data available for UNIX-like systems.)
`NetworkInterface.xml`	The IP and MAC addresses of each network interface, used to build a list of these addresses for each host. ***Note:*** *ADDM 9.0 introduced the `IPAddress` nodes which cover both IPv4 and IPv6. Prior to that, the IPv4 IP address was in the `NetworkInterface` node. This query checks both of these sources.*
`SoftwareInstance.xml`	Software installations identified by BMC Discovery's pattern language. BMC Discovery queries various properties such as processes and files of a host to determine which software is installed and its version.
`SoftwareInstanceVirtual Machine.xml`	These `SoftwareInstance` nodes are used to represent virtual machines on a host. These records are typically only created when the virtual machine is running.

Conditions for Use

Use of this executable, `FnmpADDMStage.exe`, imposes the following conditions:

- Within BMC Discovery, the XML-based API must be enabled (by default, it is enabled). BMC documentation for the XML API is available at `https://docs.bmc.com/docs/display/DISCO111/XML+API`.
- There must be HTTP or HTTPS communication available between the staging server, on which this executable runs, and the server hosting BMC Discovery.
- The staging server requires the .NET 4.5 (minimum) runtime environment.
- The staging tool must be configured with credentials that provide read access to the BMC Discovery instance. These credentials may either be configured in `FnmpADDMSettings.xml` or supplied on the command line for `FnmpADDMStage.exe` (for details see Account Configuration). For Discovery 11.1 and later, you need to grant the API access permission (api-access group), or update group permissions to include the api/access permission for the user that is specified in the `FnmpADDMSettings.xml`, who requires access to the XML Web APIs.
- For linking upstream to the staging database, you can either
  - Use a trusted connection to the SQL server and run the executable under an account that has read/write access to the staging database; or
  - Use service account credentials for SQL through a connection string, which may either be configured in `FnmpADDMSettings.xml` or supplied on the command line for `FNMPAddmStage.exe` (see Creating the Staging Database Tables for details).

Important: *It is critical that the `FnmpADDMStage.exe` tool is not run simultaneously with the `ComplianceReader` tool that uploads from the staging database to the compliance server. The adapter executable commences its writing activity with a `truncate` of the staging database to clear old results, an action which (if it occurred in the middle of an upload to the compliance server) could clearly corrupt the imported dataset. While `ComplianceReader` is blocked until `FnmpADDMStage.exe` finishes, you must adjust your schedules to prevent starting `FnmpADDMStage.exe` while the `ComplianceReader` is running.*

2

Installation and Configuration

For **on-premises** implementations, files are included in your product installation archive for the application server. As well, the latest version of the adapter is available for download from the Flexera `flexnetoperations` website. For **cloud** implementations of FlexNet Manager Suite, you also require this download: although you do not need to make changes to your central application server, there are additional components you require in the download.

You need credentials supplied by Flexera to access this download. Details of the download are included in Creating the Staging Database Tables.

- The BMC Discovery adapter suits FlexNet Manager Suite releases 9.2.3, and 2014 and later for on-premises delivery.
- The build number for this adapter is 12.0 (or higher). You can identify this number by right-clicking on `FNMPADDMStage.exe`, selecting **Properties** and looking at the **Details** tab.

Save the zipped archive to a suitable temporary location, and unzip it.

Full details of setting up the BMC Discovery adapter are included in the following sections.

Choosing a Staging Server

The FlexNet adapter for BMC Discovery requires a 'staging server' that supports installation of the adapter's executable and of the staging database. Several configurations are possible. The staging server may be installed on:

- A dedicated stand-alone server (or virtual machine)
- Any other suitable machine in your enterprise, such as a print server
- An inventory beacon
- The central application server where FlexNet Manager Suite is installed (for on-premises installations).

The requirements for a suitable server include:

- A Windows-based operating system
- Access to an installation of Microsoft Microsoft SQL Server 2012 or later, in any edition, where the staging database may be implemented (it may be on the staging server, or on a separate database server)
- The .NET 4.5 runtime environment installed

- Network access to the central application server (when not co-installed there)
- Efficient network access to each BMC Discovery server in your enterprise, using the HTTP or HTTPS protocols.

Note: *Disconnected scenarios are also possible, using the intermediate XML files saved to disk to allow manual intervention. For more information, see The Adapter Executable.*

Creating the Staging Database Tables

Once you have selected your staging server, and it can access an operating implementation of Microsoft SQL Server running a database instance you intend to use for the staging database, you should use the script provided to create the staging database and set up the appropriate database tables within it. This can be done from SQL Server Management Studio, or from the command line as described in the following procedure.

1. Download the `Adapter Tools for` FlexNet Manager Suite `2019 R2.zip` archive from the Flexera Customer Community knowledge base:

   a. Access https://flexeracommunity.force.com/customer/articles/en_US/INFO/Adapter-Tools-for-FlexNet-Manager-Suite.

   Tip: *Access requires your Customer Community user name and password. If you do not have one, use the link on the login page to request one.*

   b. Click the link `Adapter Tools for FlexNet Manager Suite`.

   A new browser tab may appear temporarily, and the download of `Adapter Tools for` FlexNet Manager Suite `2019 R2.zip` commences.

   c. In your browser dialog, choose to save the file, and if the browser allows it, direct the saved file to a convenient working location (such as `C:\temp` on a central, accessible server).

   If your browser saves the file to a default location (such as your `Downloads` folder), move or copy it to the appropriate working location when the download is finished.

2. Right-click the downloaded zip archive, and choose **Extract All...**.
3. Navigate through the unzipped archive to `Adapter Tools for` FlexNet Manager Suite `2019 R2.zip` > BMC `Atrium Discovery and Dependency Mapping Tools` > `SQL`.
4. If necessary, copy the script `ADDM_staging.sql` from the `SQL\` folder of your unzipped adapter archive to a temporary folder on your staging server.
5. Open a command prompt on the staging server.
6. In the command prompt window, execute the following command, as amended:

```
sqlcmd -S ServerName\InstanceName -i TemporaryPath\ADDM_staging.sql
```

   where:

   - The database `ADDM_Staging` is created with all necessary tables, indices, and so on.
   - *`ServerName`* is the name of the database server hosting the staging database, or its IP address, or "`.`" (dot) if

you are running the staging script on the same server as the database instance

- *InstanceName* is the name of the instance to use for the database staging tables (this parameter may be omitted if the instance is the default instance)
- *TemporaryPath* is the location where you saved the SQL procedure.

Example:

```
sqlcmd -S 192.100.0.20\Development -i C:\temp\ADDM_staging.sql
```

7. Ensure that the account under which the adapter executable will run has read/write/execute permissions on this database instance. Authentication may be through Windows NT authentication or SQL Server authentication. Using Windows NT authentication, the default account is the username running the `FnmpADDMStage.exe` adapter. SQL connection is specified as a standard connection string, which you may supply in `FnmpADDMSettings.xml`, or override with the `-c` option on the command line.

Tip: *Configuration of the account is done through SQL Server Management Studio.*

The staging database is now ready for operation.

Installing and Configuring the Staging Tool

This procedure includes many separate sub-processes to complete the set-up of the adapter executable. We start from the configuration of the inventory reader that uploads inventory gathered by the adapter.

To install and configure the staging tool:

1. Navigate to `C:\ProgramData\Flexera Software\Compliance\ImportProcedures\Inventory\Reader\`.

   This is the fixed location on the inventory beacon where the inventory reader (reaching out from the compliance server) must find configuration files to control its uploads. By default, the inventory import (starting with the reader) is triggered around 2am. Therefore you might consider scheduling this task for some time such as 10pm daily. The command line for the scheduled task (assuming that you have saved your preferred settings) is simply to invoke the executable. Any parameters not specified on the command line are taken from the settings file in the same folder as the executable.

2. From your unzipped adapter archive, copy the folder `BMC Atrium Discovery and Dependency Mapping` (found in the path `Adapter\Reader\`) to the location identified in the previous step. This folder includes at least ten XML files and a `reader.config` file. This completes configuration for the inventory reader.

3. Create a folder to contain the adapter executable and its configuration file. Location is not critical; a suggested path is under `C:\Program Files\Flexera Software`. In your chosen location, create a folder such as `ADDMAdapter`.

4. From the `FnmpADDMStage\` folder within your unzipped archive, copy both `FnmpADDMStage.exe` and `FnmpADDMSettings.xml` to your newly created folder (such as `C:\Program Files\Flexera Software\ADDMAdapter`).

5. Open your copy of `FnmpADDMSettings.xml` in a text editor of choice, and review the self-documenting comments within that file. Modify the following values as required (at the very minimum, correct the IP address of your BMC Discovery server):

   - BMC has changed the permission required to access the XML API. You need to grant the new API access permission (**api-access** group), or update group permissions to include the **api-access** permission for users who require access to the XML web API. Existing integrations will fail if you do not add the permission.
   - Update values in the first element describing the downstream connection to the BMC Discovery server, including the IP address, the account name and password for access. Keep a record of the account name and password for registering with BMC Discovery (see Account Configuration). The default values are: `<server protocol="http" address="10.200.20.138" username="exportuser" password="Pa$$w0rd" timeout="3600"/>`

     Tip: *If you do not wish to record the password in the plain text configuration file, you can use a script to retrieve the password from an encrypted store, and supply it as a command-line option when starting the `FnmpADDMStage.exe` tool.*

   - Update the second element for the connection to the staging database. The default values are: `<database connection-string="Server=.;Database=ADDM_Staging;Trusted_Connection=yes;"/>`
   - Update the third element to configure whether, and where, the executable should save XML files of the inventory collected from BMC Discovery. The default value stores any XML files below the location of the executable (but turns off storage anyway): `<staging path="." method="stream"/>` You may wish to redirect the path setting for easier access for human inspection.
   - Save your modified settings file.

6. Assuming that you do not wish to trigger the adapter manually every time it needs to run, start Windows Task Scheduler, and create a basic task to run the adapter.

   Important: *It is critical that you schedule the adapter to run at times which cannot overlap with the inventory reader uploading the results to the compliance server. Check the schedule for the compliance reader on the central compliance server, and avoid this time slot.*

   By default, the inventory import (starting with the reader) is triggered around 2am. Therefore you might consider scheduling this task for some time such as 10pm daily. The command line for the scheduled task (assuming that you have saved your preferred settings) is simply to invoke the executable. Any parameters not specified on the command line are taken from the settings file in the same folder as the executable.

This completes the configuration of the adapter executable itself. Now we can turn our attention downstream, first to possible installations on target UNIX-based machines, and then to enhancements to BMC Discovery itself.

Installing the FlexNet inventory agent

If you have chosen to install any of the UNIX-related collection patterns to enhance the inventory collection available through BMC Discovery, the FlexNet inventory agent must be copied either:

- On to each UNIX-based machine that is a target for enhanced inventory collection; or

- To a pre-configured NFS share that is accessible from each of the target UNIX-based machines.

In the following procedure, your choice of either of the above two locations is referred to as the 'target location'.

(For background information about the FlexNet inventory agent, see The FlexNet inventory agent. For choosing between the collection patterns, see Optional Patterns. Further information about deploying the FlexNet inventory agent is available in the separate PDF file *Gathering FlexNet Inventory*.)

To install the FlexNet inventory agent for UNIX collection patterns:

1. Using the installations of either FlexNet Beacon, or application server, locate the subdirectory that contains the `ndtrack` files. The default location is `C:\Program Files\Flexera Software\Inventory Beacon\RemoteExecution\Public\Inventory`
2. From this folder, copy the two files `ndtrack.sh` and `ndtrack.ini` to the target location you selected above (that is, either to a pre-configured NFS file share, or to each target UNIX-based machine). The default location pre-configured in the template file is `/opt/flexera/`, but you may modify this as required.

   Important: *The file path on all individual UNIX-based machines `must` be identical.*

3. Note the file path used (whether a file share, or the identical path used across all target devices) for entry into ADDM when you are enabling the optional collection patterns (see next section).

Configuring BMC Discovery

There are three customizations needed for BMC Discovery:

- Installing any of the optional patterns needed for the adapter in your environment (see below)
- Ensuring that BMC Discovery applies these patterns to the appropriate target computers (see Rediscovering Affected Computers)
- Ensuring that the account running the adapter has access to BMC Discovery (see Account Configuration).

Installing Optional Patterns

For information about choosing which patterns to use, see Optional Patterns. For details about enabling each of the patterns, and modifying their behavior, see Appendix A: Details of Patterns. All these patterns are contained in a single template (`Flexera.FNMP.InventoryRawData.tpl`), which must be installed first. Thereafter, individual patterns can be enabled, disabled, and modified.

1. Log in to the BMC Discovery interface, and select the **Discovery** tab.
2. Select the **Knowledge Management** button in ADDM version 10, or BMC Discovery version 11 or later (in earlier versions of ADDM, select the **Pattern Management** button).

Figure 2: The workspace in ADDM 10

3. Click on **Upload** (or in earlier versions, **Upload New Package**).

Figure 3: Choose the file from your unzipped archive

4. Click on **Choose File**, and in your unzipped archive of the adapter, select the `FlexNet Manager Platform\Installers\BMC Atrium Discovery and Dependency Mapping Tools\patterns\Flexera.FNMP.InventoryRawData.tpl` file.

5. If you are using ADDM version 8:

   a. Optionally modify (or accept) the default name of the package as `Flexera.FNMP.InventoryRawData`.

   b. Optionally, set the description to something you will find helpful, such as `FNMP export patterns`.

6. Click **Upload** (or in earlier versions, **Upload & Activate**).

   Ensure that the message `The Requested Changes were Successful` is displayed. The `Flexera.FNMP.InventoryRawData` pattern is now in the `Active` state. For further fine tuning, jump ahead to step 4 in the following procedure.

Tip: *For releases of BMC Atrium Discovery and Dependency Mapping up to and including 11.0, if you are using data imported through the BMC Discovery adapter to manage points-based licenses for IBM and Oracle, there is an additional optional pattern that collect further details about processors available on request from Flexera. Notice*

that for BMC Discovery release 11.1 or later, the additional pattern is deprecated. BMC Discovery 11.1 (and later) has improved processor information collection which makes this additional pattern unnecessary.

Enabling Optional Patterns

By default, after installation some patterns are enabled and others disabled (see Optional Patterns). The following procedure allows you to turn individual patterns on and off as required, as well as enabling (or disabling) the entire module.

1. In the user interface for BMC Discovery, select the **Discovery** tab.
2. Click **Pattern Management**, and from the list of patterns at the bottom of the page, ensure that the `Flexera.FNMP.InventoryRawData` item is enabled, and select it. (The module must be enabled before any of its member patterns can operate.)
3. From the **Pattern Package** properties, select the link for the **Pattern Module**.
4. Scroll down to the **Pattern Configuration** area.

   Several closed groups are displayed, one for each optional pattern and another for configuration of the FlexNet inventory agent on UNIX platforms.
5. For each group that you want to modify, select **Edit Configuration** to the right of the group. The group opens, and configurable controls are displayed. Select the **True** radio button to turn on a pattern, and the **False** radio button to turn off a pattern.

   More details about configuration are included in Appendix A: Details of Patterns.
6. Click **Apply**.

   The modified settings are displayed.

Tip: *If you later use the **Edit Module** button to modify the module, be sure to activate, validate, and then commit your changes.*

Rediscovering Affected Computers

Where (as is common) you are adding these patterns to an operational BMC Discovery instance that has already taken inventory of computers in your estate, you must rediscover any computers that are to be targeted through these new patterns, so that BMC Discovery applies to patterns to the rediscovered computers. To achieve this, you may either:

- Create (or wait for) a scheduled scan
- Initiate a Snapshot discovery scan
- Manually execute each pattern for suitably grouped targets, as summarized in the following procedure.

For more information about these options, see the BMC Discovery documentation available at `http://discovery.bmc.com/confluence/display/90/Documentation`.

To manually execute the new patterns:

1. In the BMC Discovery interface, select target computers (hosts or other nodes) by adding them to a group,

creating separate groups to receive distinct types of patterns (for example, one group for Windows-related patterns, and another group for UNIX-related patterns). Create your groups using either of these approaches:

- From a view node (including host) page, select **Groups** from the **Actions** list and add the node to a group.
- From a report or other search result, select the required target computers. Then, select **Groups** from the **Actions** list and add the target machines to a group.

2. From the **Discovery** tab, click **Pattern Management**.
3. Select the `Flexera.FNMP.InventoryRawData` pattern from the package list.
4. Click the **Pattern Modules** link.
5. Select the `Pattern Module` containing the pattern that you want to run.
6. Click the **Pattern** link in the heading table.
7. From the **Actions** list, select **Run Pattern**.
8. In the **Run against Group** list, select the group containing your target machines for the current pattern(s).
9. Set the `Expand` and `Execution Logging` preferences for the run.
10. Set `Additional Discovery` to **Get all new discovery data**. This forces a new discovery.

More details about this procedure are available from http://discovery.bmc.com/confluence/display/90/Manual+pattern+execution.

Account Configuration

The user name and password that the adapter needs to access the BMC Discovery API is defined in `FnmpADDMSettings.xml` (see Installing and Configuring the Staging Tool). Here we grant that account adequate rights.

1. In the BMC Discovery interface, select the **Administration** tab.
2. In the Security section of the Administration page, click the **Users** icon.
3. From the Users page, click **Add** at the bottom of the page.

   Tip: *If the account has already been registered in BMC Discovery, you can select it from the list of users and review its settings.*

4. Configure the adapter account, which can be a member of the **readonly** group. Ensure that the user name and password are exactly same values that you configured in `FnmpADDMSettings.xml`. For BMC Discovery 11.1 and later, the permission required to access the XML API has changed. You need to grant the new API access permission (**api-access** group), or update group permissions to include the **api-access** permission for users who require access to the XML Web APIs. Existing integrations will fail if you do not add the permission.

Figure 4: Sample settings for a adapter account to access BMC Discovery

5. Save your settings.

BMC Discovery is now configured for use of the adapter.

Validation and Operation

Normal operation relies on the following sequence of events:

1. BMC Discovery gathers inventory using the enhanced patterns you have enabled (details: Installing Optional Patterns, and Rediscovering Affected Computers).
2. At the time you scheduled (Installing and Configuring the Staging Tool), the adapter reads the current content of the BMC Discovery database and stages the new data in the staging database (previous data is first removed with a single `truncate` statement). The resulting status is flagged within the database.

3. Following the schedule on the central compliance server, and provided that the staging status is `Success`, the import reader uploads this content to the inventory database.
4. The next inventory import brings the final data set into the compliance database, where it is automatically taken into account for compliance calculations. As always, the inventory records must be recognized by the Application Recognition Library, and you must have the resulting application records linked to the appropriate license, for compliance calculations to proceed.

To validate operation of the adapter:

1. Wait until BMC Discovery collects new inventory, so that the enhanced collection patterns are exercised. For further details, see the BMC Discovery documentation.
2. Manually trigger the adapter executable.

   By specifying a different method parameter, you have a one-time override of the default you set in the settings XML file. For example:

   ```
   C:\Program Files\Flexera Software\ADDMAdapter\FnmpADDMStage.exe
                       -f "C:\temp" -m staged
   ```

   This will write XML files under your `C:\temp` directory for review. As well, it writes data into the staging database.
3. Inspect the saved XML files to validate the inventory gathered.
4. Use SQL Server Management Studio to validate that the data is written to the staging database. Also review the `StagingState` property in the `ADDMStagingDatabaseConfiguration` table in the staging database. Possible values are `Running`, `Failed`, or `Success`. This value must be `Success` before the BMC Discovery data can be uploaded from the staging database to the central inventory database.
5. Wait (for example, overnight) until the next inventory import and calculations have run.
6. Use FlexNet Manager Suite to validate that new evidence has been recovered. Identify which evidence has been recognized by the ARL and which new rules are required. Link the applications to appropriate licenses.

3

Known Issues

The following issue has been identified:

- UNIX-based devices that have the same host name and no detected serial number in BMC Discovery are merged into a single computer record in FlexNet Manager Suite.

4

Appendix A: Details of Patterns

While the BMC Discovery discovery tool is a state-of-the-art tool to support ITAM, the out-of-the-box collection of inventory does not capture all of the data elements required by FlexNet Manager Suite to perform an accurate software license calculation. The optional patterns in this appendix extend those data capture capacities.

Overview of Patterns

The BMC Discovery Adapter available for FlexNet Manager Suite includes six additional patterns that can be incorporated into BMC Discovery to capture these additional data attributes.

Pattern name	FlexNet inventory agent dependency	Default
`FileEvidence`	No dependency.	Tag files (all platforms): Enabled Executables (Windows only): Disabled
`InstallAnywhereEvidence`	None	Enabled
`InstallShieldMultiplatformEvidence`	None	Enabled
`UnixHardwareData`	Required	Disabled
`WindowsLastLoggedOnUser`	None	Enabled

The patterns are written in the BMC Discovery Pattern Language (TPL), for which documentation is available at `http://discovery.bmc.com/confluence/display/90/The+Pattern+Language+TPL`.

For a summary of the patterns to help decide which ones to enable or disable for your enterprise, see Optional Patterns. The following sections go into more detail about each of the patterns.

Note: *For releases of BMC Atrium Discovery and Dependency Mapping up to and including 11.0, the optional patterns listed above do not provide enough detail about processors to allow management of points-based licenses for IBM and Oracle based on data collected through BMC Discovery. If you need this capability through BMC ADDM (up to release 11.0), ask your Flexera consultant to request additional optional patterns for you. Notice that for BMC Atrium*

Discovery and Dependency Mapping release 11.1 or later, the additional pattern is deprecated. BMC Discovery 11.1 (and later) has improved processor information collection which makes this additional pattern unnecessary.

FileEvidence

Some applications cannot be recognized by installer package information alone. It is sometimes necessary to examine files that form part of the software installation, for either of two reasons:

- Some files are intended to provide identification details about an application, sometimes in human-readable formats
- Examining executable files installed with the application may help with identification, even though this is not their primary purpose.

Of the first class, identification files may take various forms. For example, many IBM applications are correctly identified by specific files that IBM installs for this purpose. Oracle and Adobe are among other publishers using specific files to identify some applications. Thus the Application Recognition Library (ARL) requires this file information to correctly identify such applications. ISO/IEC 19770-2 SWID tags are also increasingly available, and these ID tags are another useful form of identification file. The BMC Discovery Inventory Agent by default does not capture the complete set of identity files.

Secondly, in addition to gathering those specific files that identify an application (and have no other function in the application), it is sometimes necessary to identify installed executable files that are part of the application. These may be the only way to identify an application, such as a particular edition or version of a product. A standard implementation of BMC Discovery does not track executable files.

With this pattern, the functionality of BMC Discovery is extended to gather identification (tag) files on all platforms, and executable file evidence on Windows platforms. Note that the pattern does not search network shares or NFS mounts; nor does it follow symlinks (because of the risk of self-referencing loops). It also skips any files or folders that are inaccessible. Within these constraints, it provides details of files matching the specification and found within the defined paths on the local file system.

Important: *Enabling and configuring the tracking of executables within this pattern should be handled with skill and care, for two reasons:*

- *Tracking Windows executables using BMC Discovery is slow. It may be unacceptably slow to use for a wide range of directories, and you may require very targeted inventory gathering using this facility.*
- *Tracking executable files can produce a very large data set. Across large Windows server farms, the number of installation records can quickly run even into the millions, which may comprise a stress test for BMC Discovery implementations and concentrators, and (to a slightly lesser extent) for your FlexNet Manager Suite implementation.*

Note: *In FlexNet Manager Suite, you can use file evidence to track the usage of an application (perhaps with a view to reclaiming under-used licenses). While that functionality requires you to identify a watch-list of at least one executable file for each tracked application, file evidence alone is not sufficient for usage tracking: it also requires additional FlexNet agents on the managed device, and an upload path through inventory beacons that preserve the additional usage data. File evidence gathered through the BMC Discovery adaptor is not sufficient for usage tracking.*

Results

The file evidence gathered on Windows servers is somewhat richer than on UNIX-based servers.

- On both platforms, the pattern first collects the target directories from the pattern configuration file (under **Flexera.FNMP.InventoryRawData.FileEvidenceConfigs**).
- In the specified folders and their subfolders, the pattern retrieves:
  - All files with extensions listed under **File extensions to report as tag files** (assuming that **Report software tag files** has its default value of true). For each matching file found, BMC Discovery creates a `Detail` node linked to the host record. This happens for both Windows and UNIX-based systems.
  - On Windows only, and only when the setting for **Report executable files on Windows platforms** has been changed to true, files with a `.exe` extension from the same paths. For each executable file found, by default a WMI query is used to retrieve the file's name, version, and manufacturer. A `DiscoveredWMI` node in BMC Discovery is created for each WMI query (essentially for each file). However, because `DiscoveredWMI` nodes are ephemeral (may not survive future inventory gathering), the information is duplicated into a `Detail` node under the host server for each file discovered there.

By the appropriate means, then, a `Detail` node is created for each file evidence record, with the following properties dependent on the collection method:

Property	Value	Notes
`name`	File path and name	
`type`	`FNMP_FileEvidence`	
`size`	File size	
`key`	A unique key for this file, combining the values of *`name/type/host.key`*	
`version`	The release number of the file	Available only on Windows servers when executables are tracked.
`company`	The publisher of the application of which this file forms a part	Available only on Windows servers when executables are tracked.

Configuration

In the `FileEvidence` pattern, you may:

- Separately enable or disable collection of:
  - Identity tag files
  - Executable files (remember to consider the potential volume of data for this option)
- Identify the file name extensions for tag files (but there is no need to identify executable file name extensions)
- Separately for Windows and UNIX-like hosts, specify the starting point(s) in the file systems for inventory scanning to begin (searches recurse through local subdirectories).

Important: *The BMC Discovery inventory agent stops scanning at partition boundaries, even those on the local file system. This has important implications on systems, such as IBM AIX, that typically mount key paths like `/opt` on separate partitions. You must specify a search starting point within each target partition on the file system. For example, the default values of `/opt`, `/var`, and `/usr` are well suited for inventory gathering with BMC Discovery.*

All configuration items are covered in the following procedure.

To configure the FileEvidence pattern

1. In the user interface for BMC Discovery, select the **Discovery** tab.
2. Click **Pattern Management**, and from the list of patterns at the bottom of the page, ensure that your current `Flexera.FNMP.InventoryRawData` item is enabled, and select it. (The module must be enabled before any of its member patterns can operate.)
3. From the **Pattern Package** properties, select the link for the **Pattern Module**.
4. Scroll down to the **Pattern Configuration** area.
5. In the **FileEvidence** group, click **Edit Configuration**.
6. Select the appropriate true/false radio button for **Report software tag files**. If you are turning off the collection of tag files, skip the next step.
7. Adjust the list of **File extensions to report as tag files**, if necessary deleting values or adding new ones that you have identified. Each extension must stand alone on its own line.
8. Select the appropriate true/false radio button for **Report executable files**. Review the discussion before this procedure while considering the data quantities implied by enabling this option.

   If you have now turned off both options, so that both options now have **False** selected, you have completed the process, and may skip the next step. If either option has **True** selected, continue to define the paths for the inventory gathering.

   The configuration changes are saved.
9. For both types of operating system, customize the ...**scan these file paths for evidence** list.
   - Each file path must stand alone on its own line, and must be absolute (starting from root).
   - For Windows, the path must include the drive letter with its colon delimiter.
   - These same paths are scanned for identity (tag) files and for executable files.
10. Click **Apply** (at the bottom of this group).

    The configuration changes are saved.

InstallAnywhereEvidence

InstallAnywhere is a package installation solution developed by Flexera. Packages track their installation status in an XML file, which is interrogated by this pattern.

After collection of the inventory data, a `Detail` node is linked to the host computer for each package installed on the computer:

Property	Value
`name`	The name of the application as identified by InstallAnywhere.
`type`	`FNMP_InstallerEvidence`
`vendor`	The publisher of the application
`version`	The release number of the application.
`install_date`	The date that the application was installed on this host computer.
`evidence`	`IA` (fixed string literal).
`key`	A unique key for this installation record, combining (with the different literal text separators shown) the values of *`name:version/type:evidence/host.key`*

To configure the InstallAnywhereEvidence pattern:

1. In the user interface for BMC Discovery, select the **Discovery** tab.
2. Click **Pattern Management**, and from the list of patterns at the bottom of the page, ensure that your current `Flexera.FNMP.InventoryRawData` item is enabled, and select it. (The module must be enabled before any of its member patterns can operate.)

   Tip: *The `InstallAnywhereEvidence` pattern is enabled by default when the pattern collection is initially enabled. If it has previously been disabled, you can re-enable it with the remainder of this procedure.*

3. From the **Pattern Package** properties, select the link for the **Pattern Module**.
4. Scroll down to the **Pattern Configuration** area.
5. In the **InstallAnywhereEvidence** group, click **Edit Configuration**.
6. Select the **True** radio button for **Report installations by InstallAnywhere**.
7. Click **Apply** (at the bottom of this group).

InstallShieldMultiplatformEvidence

InstallShield Multiplatform is an older packaging solution from Flexera, in use by many software publishers. InstallShield stores software information in “vital product data” (VPD) collections, which were originally stored in flat files and subsequently in SQL scripts.

This pattern locates either format of VPD storage (which may exist in a range of locations across different platforms), extracts the data, and creates a `Detail` node for the software installation linked to the host computer:

Property	Value
`name`	The name of the application as identified by InstallShield.

Property	Value
`type`	`FNMP_InstallerEvidence`
`vendor`	The publisher of the application
`version`	The release number of the application.
`install_date`	The date that the application was installed on this host computer.
`evidence`	`ISMP` (fixed string literal).
`product_code`	The product identification code recorded (usually) by the publisher for the particular application. This is not standardized and may be used as the publisher desires.
`key`	A unique key for this installation record, combining (with the different literal text separators shown) the values of *`name:version/type:evidence/host.key`*

To configure the `InstallShieldMultiplatformEvidence` pattern:

1. In the user interface for BMC Discovery, select the **Discovery** tab.
2. Click **Pattern Management**, and from the list of patterns at the bottom of the page, ensure that your current `Flexera.FNMP.InventoryRawData` item is enabled, and select it. (The module must be enabled before any of its member patterns can operate.)

   Tip: *The `InstallShieldMutiplatformEvidence` pattern is enabled by default when the pattern collection is initially enabled. If it has previously been disabled, you can re-enable it with the remainder of this procedure.*

3. From the **Pattern Package** properties, select the link for the **Pattern Module**.
4. Scroll down to the **Pattern Configuration** area.
5. In the **InstallShieldMultiplatformEvidence** group, click **Edit Configuration**.
6. Select the **True** radio button for **Report installations by InstallShield Multiplatform**.
7. Click **Apply** (at the bottom of this group).

UnixHardwareData

Especially for managing the corporate Data Centre, it is critical for FlexNet Manager Suite to have accurate hardware inventory for capacity-based license metrics (Processor, Core, IBM PVU, and so on). While BMC Discovery can capture this information for a Windows server, it may not consistently capture it for servers running UNIX or Linux. For example, BMC Discovery does not currently report:

- CPUs and cores on Linux, nor cores on virtual machines
- LPARs on IBM AIX (correctly)
- Solaris resource pools.

This pattern retrieves hardware data on UNIX-based systems using the FlexNet inventory agent.

Note: *The pattern must be configured with the installed location of the agent. The agent may be installed either in the same location on the file system of each target UNIX server, or on a file share accessible to all target devices. This is included in the configuration process described below.*

It executes the inventory agent, which writes the collected data to a file in the `/var/tmp/flexera/addm/` folder on the host server. BMC Discovery then reads this output, and for each installed file, creates a `Detail` node linked to the host record:

Property	Value
`Name`	`FNMP hardware evidence for %host.name%`
`Type`	`FNMP_HardwareEvidence`
`Key`	`%type%/%host.key%`

Property	Value
(Additional properties)	Each records one of the following hardware properties: • Disk size • IP Address • MAC Address • Model • Number of cores • Number of disks • Number of logical processors • Number of processors • OS • Processor speed • Processor type • RAM (total physical memory) • Vendor. If the machine is found to be a virtual machine, the following additional properties are collected: • Node capacity • Node capacity in cores • Node capacity in threads • Physical shared pool capacity • Physical shared pool capacity in cores • Physical shared pool ID • Shared pool capacity • Shared pool capacity in cores • Shared pool ID • VM capacity • VM capacity in cores • VM entitlement • VM ID

Property	Value
	• VM is capped • VM is shared type • VM name • VM type

To configure the `UnixHardware` pattern:

1. In the user interface for BMC Discovery, select the **Discovery** tab.
2. Click **Pattern Management**, and from the list of patterns at the bottom of the page, ensure that your current `Flexera.FNMP.InventoryRawData` item is enabled, and select it. (The module must be enabled before any of its member patterns can operate.)
3. From the **Pattern Package** properties, select the link for the **Pattern Module**.
4. Scroll down to the **Pattern Configuration** area.
5. In the **Flexera.FNMP.InventoryRawData.CommonFlexeraInventoryAgentConfigs** group, click **Edit Configuration**.
6. Enter the path (only, not including the file name) to the inventory agent executable. (For details about installing the FlexNet inventory agent, see Installing the FlexNet inventory agent.)
7. Click **Apply** (at the bottom of this group).
8. In the **UnixHardware** group, click **Edit Configuration**.
9. Select the **True** radio button for **Report UNIX hardware properties**.
10. Click **Apply** (at the bottom of this group).

WindowsLastLoggedOnUser

For a Windows-based computer, standard BMC Discovery collection does not capture any inventory related to the Windows Logon. User identification is important for FlexNet Manager Suite to accurately calculate license consumption for user-based licensed, such as a Named User license. (There is no equivalent data for UNIX-like systems available through the BMC Discovery adapter.)

This pattern queries the registry at `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Authentication\LogonUI\LastLoggedOnUser` to find the last logged-on user (for legacy Windows platforms before Vista, it queries WMI for the value of `UserName` from `Win32_ComputerSystem`). If the query is successful, a `DiscoveredRegistryValue` node is created for the host computer:

Property	Value
`Name`	The user name.
`Type`	`FNMP_LastLoggedOnUser`

Property	Value
Key	%type%/%host.key%

To configure the `WindowsLastLoggedOnUser` pattern:

1. In the user interface for BMC Discovery, select the **Discovery** tab.
2. Click **Pattern Management**, and from the list of patterns at the bottom of the page, ensure that your current `Flexera.FNMP.InventoryRawData` item is enabled, and select it. (The module must be enabled before any of its member patterns can operate.)

   Tip: *The `WindowsLastLoggedOnUser` pattern is enabled by default when the pattern collection is initially enabled. If it has previously been disabled, you can re-enable it with the remainder of this procedure.*

3. From the **Pattern Package** properties, select the link for the **Pattern Module**.
4. Scroll down to the **Pattern Configuration** area.
5. In the **WindowsLastLoggedOnUser** group, click **Edit Configuration**.
6. Select the **True** radio button for **Report Windows last logged-on user**.
7. Click **Apply** (at the bottom of this group).

5

Appendix B: Data Mappings

This appendix maps the data transformations from BMC Discovery to FlexNet Manager Suite in two stages:

- From the source nodes in BMC Discovery datastore to the staging database (suggested: `ADDM_Stage`). The data constructs within this database are mirrored in the XML files that may be saved on the staging server for your inspection. For this reason, this section may also be used as a guide to understanding the sources of data within BMC Discovery that populate the various XML entities.
- From the staging database (suggested: `ADDM_Stage`) to the final destination within the web interface of FlexNet Manager Suite. This is not a simple and direct path: the `ComplianceReader` component first loads the data from the staging database to various compliance database tables with names starting with `Imported`, where the data waits until the next "data import" (usually associated with a full license consumption calculation, by default scheduled overnight). In effect, the data first moves from remote staging tables to local staging tables. Thereafter the data is normalized and loaded into a range of compliance database tables, for the most part arrayed around the `ComplianceComputer` table. Since navigating these tables is not straight-forward, we here present the data in its more visible form, within the web interface. Serious devotees of database structures can find all compliance database tables documented in the *FlexNet Manager Suite Schema Reference* PDF file, available through the title page of online help.

In both parts, each destination has its own topic, so that you can jump to destination names using the table of contents of this PDF (in the first of these sections, the destinations are table names within the staging database; and in the second, the destinations are "object" names presented in the web interface).

Because an individual node in BMC Discovery may link to multiple staging tables, it's not quite so easy to find the original nodes in BMC Discovery. One way is to do a search in the PDF file for "SEARCH *`nodeName`*", since each staging database topic begins with the Query Language statements used to extract the related data from BMC Discovery.

Source to Staging

This section provides:

- The Query Language queries exercised against BMC Discovery to extract data
- The tables in the staging database, and within each table the individual properties (database columns), where the data is staged.

For each table topic, the following details are included:

- The source alias (usually identical to the following)
- The property/column name in the staging table
- The data type in the staging table
- For string types, the maximum number of characters (except as noted, excess string length is truncated).

Each topic also notes the name of the XML file that may be saved on the staging server (depending on the command line given to the staging tool).

ADDMVersion_ci (Staging Table)

Data in this staging table may also be saved in `ADDMVersion.xml` on the staging server.

BMC Discovery query

This query extracts the version information for BMC Discovery, saving the details in the table below:

```
LOOKUP Version
```

Mapping into ADDMVersion_ci

Source alias	ADDMVersion_ci property	Type	Max
Version	Version	varchar	32
Release	Release	varchar	32
Release Name	Release Name	varchar	32

Cluster_ci (Staging Table)

Data in this staging table may also be saved in `Cluster.xml` on the staging server.

BMC Discovery query

This query extracts data from the `Cluster` node in the BMC Discovery datastore, and provides the aliases for properties referenced in the table below:

```
SEARCH Cluster
   SHOW
    key AS 'ClusterKey',
    type AS 'ClusterType',
    (type = 'vCenter Cluster' AND
     (single(#ManagedElement:Management:Manager:SoftwareInstance.instance)
          + '|' + id)
     OR NONE) AS 'InternalIdentifier',
```

```
    (cluster_name OR name) AS 'InternalName',
    single(#HostContainer:HostContainment:ContainedHost:Host.domain)
          AS 'Domain',
    formatTime(modified(#), "%FT%T") AS 'LastUpdate'
```

Mapping into Cluster_ci

Source alias	Cluster_ci property	Type	Max
ClusterKey	ClusterKey	nvarchar	400
ClusterType	ClusterType	nvarchar	255
Domain	Domain	nvarchar	512
InternalIdentifier	InternalIdentifier	nvarchar	512
InternalName	InternalName	nvarchar	512
LastUpdate	LastUpdate	datetime	

ClusterHost_ci (Staging Table)

Data in this staging table may also be saved in `ClusterHost.xml` on the staging server.

BMC Discovery query

This query extracts data from the `Cluster` node in the BMC Discovery datastore, and provides the aliases for properties referenced in the table below:

```
SEARCH Cluster
   WHERE (NODECOUNT
         (TRAVERSE HostContainer:HostContainment:ContainedHost:Host) > 0)
   SHOW
    key AS 'ClusterKey',
    EXPLODE #HostContainer:HostContainment:ContainedHost:Host.key
         AS 'HostKey'
```

Mapping into ClusterHost_ci

Source alias	ClusterHost_ci property	Type	Max
ClusterKey	ClusterKey	nvarchar	400
HostKey	HostKey	varchar	64

CPUInformationDetail_ci (Staging Table)

Data in this staging table may also be saved in `CPUInformationDetail.xml` on the staging server.

BMC Discovery query

This query extracts data from the `Detail` node in the BMC Discovery datastore, and provides the aliases for properties referenced in the table below.

Tip: *This data relies on a custom CPU pattern available through your Flexera consultant. Be aware that from BMC Discovery release 11.1, BMC provided improved data gathering with new `ProcessorInfo` functionality. However, this does not gather the full dataset enabled through the custom pattern, and it remains best practice to combine the data gathered by BMC Discovery with additional data available through the custom pattern.*

```
SEARCH Detail
   WHERE (NODECOUNT(TRAVERSE Detail:Detail:ElementWithDetail:Host) > 0)
    AND (type = 'CPU Information')
   SHOW
    host_partitioning_type AS 'HostType',
    (
     (host_partitioning_type = 'physical' AND total_logical_cpus
      OR allocated_threads) OR none
    ) AS 'LogicalProcessors',
    (
     (host_partitioning_type = 'physical' AND total_cpu_cores
      OR involved_cores) OR none
    ) AS 'Cores',
    (
     (host_partitioning_type = 'physical' AND total_cpu_sockets
      OR involved_sockets) OR none
    ) AS 'Processors',
    (total_logical_cpus OR none) AS 'PhysicalLogicalProcessors',
    (total_cpu_cores OR none) AS 'PhysicalCores',
    (total_cpu_sockets OR none) AS 'PhysicalProcessors',
    zone_pool_name AS 'ZonePoolName',
    lpar_id_to_partition AS 'LPARIdToPartition',
    lpar_id_to_system AS 'LPARIdToSystem',
    (
     (lpar_partition_number <> '-' AND lpar_partition_number) OR none
     ) AS 'LPARPartitionNumber',
    lpar_partition_name AS 'LPARPartitionName',
    lpar_type AS 'LPARType',
    lpar_mode AS 'LPARMode',
    (lpar_active_physical_cpus_in_system OR none)
        AS 'LPARActivePhysicalCpusInSystem',
    (lpar_shared_physical_cpus_in_system OR none)
        AS 'LPARSharedPhysicalCpusInSystem',
    ((lpar_shared_pool_id <> '-' AND lpar_shared_pool_id) OR none)
```

```
        AS 'LPARSharedPoolID',
    (lpar_active_cpus_in_pool OR none) AS 'LPARActiveCpusInPool',
    (lpar_entitled_capacity OR none) AS 'LPAREntitledCapacity',
    #Detail:Detail:ElementWithDetail:Host.key AS 'HostKey'
```

Mapping into CPUInformationDetail_ci

Source alias	CPUInformationDetail_ci property	Type	Max
Cores	Cores	int	
HostKey	HostKey	varchar	64
HostType	HostType	varchar	32
LogicalProcessors	LogicalProcessors	int	
LPARActiveCpusInPool	LPARActiveCpusInPool	int	
LPARActivePhysical CpusInSystem	LPARActivePhysical CpusInSystem	int	
LPAREntitledCapacity	LPAREntitledCapacity	float	
LPARIdToPartition	LPARIdToPartition	nvarchar	200
LPARIdToSystem	LPARIdToSystem	nvarchar	200
LPARMode	LPARMode	nvarchar	255
LPARPartitionName	LPARPartitionName	nvarchar	255
LPARPartitionNumber	LPARPartitionNumber	int	
LPARSharedPhysical CpusInSystem	LPARSharedPhysical CpusInSystem	int	
LPARSharedPoolID	LPARSharedPoolID	int	
LPARType	LPARType	nvarchar	255
PhysicalCores	PhysicalCores	int	
PhysicalLogicalProcessors	PhysicalLogicalProcessors	int	
PhysicalProcessors	PhysicalProcessors	int	
Processors	Processors	int	
ZonePoolName	ZonePoolName	nvarchar	200

DiscoveredPackages_ci (Staging Table)

Data in this staging table may also be saved in `DiscoveredPackages.xml` on the staging server.

BMC Discovery query

This query extracts data from the `Host` node in the BMC Discovery datastore, and provides the aliases for properties referenced in the table below:

```
SEARCH Host
  STEP IN Host:HostedSoftware:InstalledSoftware:Package
  SHOW
   #:InstalledSoftware:Package.name AS 'Name',
   #:InstalledSoftware:Package.version AS 'Version',
   #:InstalledSoftware:Package.vendor AS 'Vendor',
   #:Host:Host.key as 'HostKey'
```

Mapping into DiscoveredPackages_ci

Source alias	DiscoveredPackages_ci property	Type	Max
HostKey	HostKey	varchar	64
Name	Package	nvarchar	255
Vendor	Vendor	nvarchar	255
Version	Version	nvarchar	255

DiscoveredService_ci (Staging Table)

Data in this staging table may also be saved in `DiscoveredService.xml` on the staging server.

BMC Discovery query

This query extracts data from the `Host` node in the BMC Discovery datastore, and provides the aliases for properties referenced in the table below:

```
SEARCH Host
   TRAVERSE InferredElement:Inference:Associate:DiscoveryAccess
    WHERE end_state = 'GoodAccess'
   TRAVERSE DiscoveryAccess:DiscoveryAccessResult:DiscoveryResult:ServiceList
   TRAVERSE List:List:Member:DiscoveredService
    WHERE name = 'vmms' AND display_name = 'Hyper-V Virtual Machine Management'
   SHOW
    name AS 'Name',
    #Member:List:List:ServiceList.#DiscoveryResult:DiscoveryAccessResult:
            DiscoveryAccess:DiscoveryAccess.#Associate:Inference:
            InferredElement:Host.key AS 'HostKey'
```

Mapping into DiscoveredService_ci

Source alias	DiscoveredService_ci property	Type	Max
HostKey	HostKey	varchar	64
Name	Name	nvarchar	255

DiscoveredVirtualMachine_ci (Staging Table)

Data in this staging table may also be saved in `DiscoveredVirtualMachine.xml` on the staging server.

BMC Discovery query

This query extracts data from the `Host` node in the BMC Discovery datastore, and provides the aliases for properties referenced in the table below:

```
SEARCH Host
    TRAVERSE InferredElement:Inference:Associate:DiscoveryAccess
     WHERE end_state = 'GoodAccess'
    TRAVERSE DiscoveryAccess:DiscoveryAccessResult:DiscoveryResult:VirtualMachineList
     WHERE access_method IS DEFINED
    TRAVERSE List:List:Member:DiscoveredVirtualMachine
   SHOW
    name AS 'Name',
    uuid AS 'UUID',
    config_os_full AS 'ConfigOSFull',
    power_state AS 'PowerState',
    vsphere_id AS 'vSphereID',
    #Member:List:List:VirtualMachineList.access_method AS 'AccessMethod',
    formatTime(modified(#), "%FT%T") AS 'LastUpdate',
    #Member:List:List:VirtualMachineList.#DiscoveryResult:DiscoveryAccessResult:
        DiscoveryAccess:DiscoveryAccess.#Associate:
        Inference:InferredElement:Host.key AS 'HostKey'
```

Mapping into DiscoveredVirtualMachine_ci

Source alias	DiscoveredVirtualMachine_ci property	Type	Max
AccessMethod	AccessMethod	nvarchar	255
ConfigOSFull	ConfigOSFull	nvarchar	512
HostKey	HostKey	varchar	64
LastUpdate	LastUpdate	datetime	
Name	Name	nvarchar	512

Source alias	DiscoveredVirtualMachine_ci property	Type	Max
PowerState	PowerState	nvarchar	255
UUID	UUID	nvarchar	255
vSphereID	vSphereID	nvarchar	255

FileEvidenceDetail_ci (Staging Table)

Data in this staging table may also be saved in `FileEvidenceDetail.xml` on the staging server.

BMC Discovery query

This query extracts data from the `Detail` node in the BMC Discovery datastore, and provides the aliases for properties referenced in the table below:

```
SEARCH Detail
   WHERE (NODECOUNT(TRAVERSE Detail:Detail:ElementWithDetail:Host) > 0)
    AND (type = 'FNMP_FileEvidence')
   SHOW name AS 'FileName',
    size AS 'FileSize',
    description AS 'Description',
    company AS 'Company',
    version AS 'FileVersion',
    #Detail:Detail:ElementWithDetail:Host.key AS 'HostKey'
```

Mapping into FileEvidenceDetail_ci

Source alias	FileEvidenceDetail_ci property	Type	Max
Company	Company	nvarchar	255
Description	Description	nvarchar	255
FileName	FileName	nvarchar	800
FileSize	FileSize	bigint	
FileVersion	FileVersion	nvarchar	255
HostKey	HostKey	varchar	64

FileSystem_ci (Staging Table)

Data in this staging table may also be saved in `FileSystem.xml` on the staging server.

BMC Discovery query

This query extracts data from the `FileSystem` node in the BMC Discovery datastore, and provides the aliases for properties referenced in the table below:

```
SEARCH FileSystem
   WHERE (NODECOUNT
          (TRAVERSE MountedFileSystem:FileSystemMount:Mounter:Host) > 0)
    AND fs_kind = 'LOCAL'
    AND size IS DEFINED
    AND size > 0
    AND fs_type <> 'tmpfs'
    AND fs_type <> 'devtmpfs'
    AND fs_type <> 'DevFS'
   SHOW
    fs_name AS 'Name',
    size AS 'Size',
    #MountedFileSystem:FileSystemMount:Mounter:Host.key AS 'HostKey'
```

Mapping into FileSystem_ci

Source alias	FileSystem_ci property	Type	Max
HostKey	HostKey	varchar	64
Name	Name	nvarchar	255
Size	Size	bigint	

HardwareEvidenceDetail_ci (Staging Table)

Data in this staging table may also be saved in `HardwareEvidenceDetail.xml` on the staging server.

This data is generated through use of the optional `UnixHardwareData` pattern, which triggers the Flexera inventory component `ndtrack` ("the tracker"). Before staging, the information is visible in the `Detail` node of BMC Discovery, with a type of `FNMP_HardwareEvidence`.

BMC Discovery query

This query extracts data from the `Detail` node in the BMC Discovery datastore, and provides the aliases for properties referenced in the table below:

```
SEARCH Detail
   WHERE (NODECOUNT(TRAVERSE Detail:Detail:ElementWithDetail:Host) > 0)
    AND (type = 'FNMP_HardwareEvidence')
   SHOW
    os AS 'OS',
    num_processors AS 'Processors',
    num_cores AS 'Cores',
```

```
    num_logical_processors AS 'LogicalProcessors',
    processor_type AS 'ProcessorType',
    processor_speed AS 'ProcessorSpeed',
    ram AS 'RAM',
    vendor AS 'Vendor',
    model AS 'Model',
    ip_address AS 'IPAddress',
    mac_address AS 'MACAddress',
    num_disks AS 'NumDisks',
    disk_size AS 'DiskSize',
    vm_type AS 'VMType',
    node_capacity AS 'NodeCapacity',
    node_capacity_in_cores AS 'NodeCapacityInCores',
    node_capacity_in_threads AS 'NodeCapacityInThreads',
    physical_shared_pool_capacity AS 'PhysicalSharedPoolCapacity',
    physical_shared_pool_capacity_in_cores
       AS 'PhysicalSharedPoolCapacityInCores',
    physical_shared_pool_id AS 'PhysicalSharedPoolID',
    shared_pool_capacity AS 'SharedPoolCapacity',
    shared_pool_capacity_in_cores AS 'SharedPoolCapacityInCores',
    shared_pool_id AS 'SharedPoolID',
    vm_capacity AS 'VMCapacity',
    vm_capacity_in_cores AS 'VMCapacityInCores',
    vm_entitlement AS 'VMEntitlement',
    vm_id AS 'VMID',
    vm_is_capped AS 'VMIsCapped',
    vm_is_shared_type AS 'VMIsSharedType',
    vm_name AS 'VMName',
    partition_id AS 'PartitionID',
    partition_number AS 'PartitionNumber',
    #Detail:Detail:ElementWithDetail:Host.key AS 'HostKey'
```

Mapping into HardwareEvidenceDetail_ci

Source alias	HardwareEvidenceDetail_ci property	Type	Max
Cores	Cores	smallint	
DiskSize	DiskSize	bigint	
HostKey	HostKey	varchar	64
IPAddress	IP	nvarchar	512
LogicalProcessors	LogicalProcessors	smallint	
MACAddress	MAC	nvarchar	512
Model	Model	nvarchar	512

Source alias	HardwareEvidenceDetail_ci property	Type	Max
NodeCapacity	NodeCapacity	smallint	
NodeCapacityInCores	NodeCapacityInCores	smallint	
NodeCapacityInThreads	NodeCapacityInThreads	smallint	
NumDisks	NumDisks	int	
OS	OS	nvarchar	256
PartitionID	PartitionID	nvarchar	512
PartitionNumber	PartitionNumber	int	
PhysicalShared PoolCapacity	PhysicalShared PoolCapacity	smallint	
PhysicalSharedPool CapacityInCores	PhysicalSharedPool CapacityInCores	smallint	
PhysicalSharedPoolID	PhysicalSharedPoolID	nvarchar	255
Processors	Processors	smallint	
ProcessorSpeed	ProcessorSpeed	int	
PhysicalLogicalProcessors	PhysicalLogicalProcessors	int	
ProcessorType	ProcessorType	nvarchar	128
RAM	RAM	bigint	
SharedPoolCapacity	SharedPoolCapacity	smallint	
SharedPoolCapacityInCores	SharedPoolCapacityInCores	smallint	
SharedPoolID	SharedPoolID	nvarchar	255
Vendor	Vendor	nvarchar	255
VMCapacity	VMCapacity	float	
VMCapacityInCores	VMCapacityInCores	float	
VMEntitlement	VMEntitlement	float	
VMID	VMID	nvarchar	255
VMIsCapped	VMIsCapped	bit	
VMIsSharedType	VMIsSharedType	bit	
VMName	VMName	nvarchar	512
VMType	VMType	nvarchar	128

Host_ci (Staging Table)

Data in this staging table may also be saved in `Host.xml` on the staging server.

BMC Discovery query

This query extracts data from the `Host` node in the BMC Discovery datastore, and provides the aliases for properties referenced in the table below.

Tip: *The following properties are recent additions to BMC Discovery:*

- *`num_cores` is available from BMC Discovery release 11.0*
- *`zone_pool_name` is available from BMC Discovery release 11.1.*

```
SEARCH Host
  SHOW
   hostname AS 'Hostname',
   (dns_domain OR domain) AS 'Domain',
   os AS 'OS',
   os_type AS 'OSType',
   os_class AS 'OSClass',
   service_pack AS 'ServicePack',
   uuid AS 'UUID',
   (
       num_processors
       OR nodecount(traverse Host:Detail:Hardware:ProcessorInfo)
   ) AS 'Processors',
   (
       num_cores
       OR ((num_processors > 0 AND cores_per_processor > 0)
           AND (num_processors * cores_per_processor))
       OR ((num_logical_processors > 0 AND threads_per_core > 0)
           AND (num_logical_processors/threads_per_core))
       OR none
   ) AS 'Cores',
   num_logical_processors AS 'LogicalProcessors',
   processor_type AS 'ProcessorType',
   processor_speed AS 'ProcessorSpeed',
   ram AS 'RAM',
   vendor AS 'Vendor',
   model AS 'Model',
   serial AS 'Serial',
   hostid AS 'HostID',
   partition_id AS 'PartitionID',
   lpar_name AS 'LPARName',
   lpar_partition_number AS 'LPARNumber',
   wparid AS 'WPARID',
   npar_partition_number AS 'NPARNumber',
```

```
    vpar_partition_number AS 'VPARNumber',
    ldom_name AS 'LDOMName',
    ldom_role AS 'LDOMRole',
    zonename AS 'Zonename',
    zone_uuid AS 'ZoneUUID',
    (zone_pool_name OR none) AS 'ZonePoolName',
    formatTime(modified(#), "%FT%T") as 'ModifiedDate',
    key AS 'HostKey
```

Mapping into Host_ci

Source alias	Host_ci property	Type	Max
Cores	Cores	smallint	
Domain	Domain	nvarchar	255
HostID	HostID	nvarchar	255
HostKey	HostKey	varchar	64
Hostname	Hostname	nvarchar	255
LDOMName	LDOMName	nvarchar	255
LDOMRole	LDOMRole	nvarchar	255
LogicalProcessors	LogicalProcessors	smallint	
LPARName	LPARName	nvarchar	255
LPARNumber	LPARNumber	int	
Model	Model	nvarchar	512
ModifiedDate	ModifiedDate	datetime	
NPARNumber	NPARNumber	nvarchar	64
OS	OS	nvarchar	1024
OSClass	OSClass	nvarchar	64
OSType	OSType	nvarchar	255
PartitionID	PartitionID	nvarchar	255
Processors	Processors	smallint	
ProcessorSpeed	ProcessorSpeed	int	
ProcessorType	ProcessorType	nvarchar	128
RAM	RAM	bigint	
Serial	Serial	nvarchar	255

Source alias	Host_ci property	Type	Max
ServicePack	ServicePack	nvarchar	128
UUID	UUID	nvarchar	255
Vendor	Vendor	nvarchar	255
VPARNumber	VPARNumber	nvarchar	64
WPARID	WPARID	int	
Zonename	Zonename	nvarchar	255
ZonePoolName	ZonePoolName	nvarchar	255
ZoneUUID	ZoneUUID	nvarchar	255

HostContainerProcessorInfo_ci (Staging Table)

This data is available only from ADDM 11.0 and later releases.

Data in this staging table may also be saved in `HostContainerProcessorInfo.xml` on the staging server.

BMC Discovery query

This query extracts data from the `ProcessorInfo` node in the BMC Discovery datastore, and provides the aliases for properties referenced in the table below:

```
SEARCH ProcessorInfo
  WHERE NODECOUNT(TRAVERSE Hardware:Detail:HostContainer:HostContainer) > 0
  SHOW
    #Hardware:Detail:HostContainer:HostContainer.num_logical_processors
         AS 'LogicalProcessors',
    #Hardware:Detail:HostContainer:HostContainer.num_cores AS 'Cores',
    #Hardware:Detail:HostContainer:HostContainer.num_processors
         AS 'Processors',
    EXPLODE #Hardware:Detail:HostContainer:HostContainer.#HostContainer:
         HostContainment:ContainedHost:Host.#InferredElement:Inference:Associate:
         DiscoveryAccess.#DiscoveryAccess:DiscoveryAccessResult:DiscoveryResult:
         HostInfo.systemid AS 'SystemID',
    EXPLODE #Hardware:Detail:HostContainer:HostContainer.#HostContainer:
         HostContainment:ContainedHost:Host.hostid AS 'HostID',
    #Hardware:Detail:HostContainer:HostContainer.#HostContainer:HostContainment:
         ContainedHost:Host.vendor AS 'Vendor'
```

Mapping into HostContainerProcessorInfo_ci

Source alias	HostContainerProcessorInfo_ci property	Type	Max
Cores	Cores	smallint	
HostID	HostID	nvarchar	255
LogicalProcessors	LogicalProcessors	smallint	
Processors	Processors	smallint	
SystemID	SystemID	nvarchar	255
Vendor	Vendor	nvarchar	255

HostDetail_ci (Staging Table)

This data is available only from BMC Discovery 11.1 and later releases. The latest Technology Knowledge Update must be applied to the BMC Discovery 11.1 release.

Data in this staging table may also be saved in `HostDetail.xml` on the staging server.

BMC Discovery query

This query extracts data from the `Detail` node in the BMC Discovery datastore, and provides the aliases for properties referenced in the table below:

```
SEARCH Detail
  WHERE type = 'LPAR Resource'
  SHOW
    id_to_partition as 'LPARIdToPartition',
    id_to_system as 'LPARIdToSystem',
    #Detail:Detail:ElementWithDetail:Host.key as 'HostKey'
```

Mapping into HostDetail_ci

Source alias	HostDetail_ci property	Type	Max
HostKey	HostKey	varchar	64
LPARIdToPartition	LPARIdToPartition	nvarchar	200
LPARIdToSystem	LPARIdToSystem	nvarchar	200

HostInfo_ci (Staging Table)

Data in this staging table may also be saved in `HostInfo.xml` on the staging server.

BMC Discovery query

This query extracts data from the `Host` node in the BMC Discovery datastore, and provides the aliases for properties referenced in the table below:

```
SEARCH Host
  TRAVERSE InferredElement:Inference:Associate:DiscoveryAccess
   WHERE end_state = 'GoodAccess'
  TRAVERSE DiscoveryAccess:DiscoveryAccessResult:DiscoveryResult:HostInfo
   WHERE NODECOUNT(TRAVERSE Primary:Inference:InferredElement:Host) > 0
  SHOW
   systemid AS 'SystemID',
   lpar_partition_number AS 'LPARPartitionNumber',
   lpar_partition_name AS 'LPARPartitionName',
   lpar_type AS 'LPARType',
   lpar_mode AS 'LPARMode',
   lpar_active_physical_cpus_in_system AS 'LPARActivePhysicalCpusInSystem',
   lpar_shared_physical_cpus_in_system AS 'LPARSharedPhysicalCpusInSystem',
   lpar_shared_pool_id AS 'LPARSharedPoolID',
   lpar_active_cpus_in_pool AS 'LPARActiveCpusInPool',
   lpar_entitled_capacity AS 'LPAREntitledCapacity',
   formatTime(#DiscoveryResult:DiscoveryAccessResult:DiscoveryAccess:
              DiscoveryAccess.starttime,"%FT%T") as 'InventoryDate',
   #Primary:Inference:InferredElement:Host.key AS 'HostKey'
```

Mapping into HostInfo_ci

Source alias	HostInfo_ci property	Type	Max
HostKey	HostKey	varchar	64
InventoryDate	InventoryDate	datetime	
LPARActiveCpusInPool	LPARActiveCpusInPool	int	
LPARActivePhysical CpusInSystem	LPARActivePhysical CpusInSystem	int	
LPAREntitledCapacity	LPAREntitledCapacity	float	
LPARMode	LPARMode	nvarchar	255
LPARPartitionName	LPARPartitionName	nvarchar	255
LPARPartitionNumber	LPARPartitionNumber	int	
LPARSharedPhysical CpusInSystem	LPARSharedPhysical CpusInSystem	int	
LPARSharedPoolID	LPARSharedPoolID	int	
LPARType	LPARType	nvarchar	255

Source alias	HostInfo_ci property	Type	Max
SystemID	SystemID	nvarchar	255

InstallerEvidenceDetail_ci (Staging Table)

Data in this staging table may also be saved in `InstallerEvidenceDetail.xml` on the staging server.

BMC Discovery query

This query extracts data from the `Detail` node in the BMC Discovery datastore, and provides the aliases for properties referenced in the table below:

```
SEARCH Detail
 WHERE (NODECOUNT(TRAVERSE Detail:Detail:ElementWithDetail:Host) > 0)
  AND (type = 'FNMP_InstallerEvidence')
 SHOW name AS 'Name',
  version AS 'Version',
  vendor AS 'Vendor',
  install_date AS 'InstallDate',
  evidence AS 'Evidence',
  product_code AS 'ProductCode',
  #Detail:Detail:ElementWithDetail:Host.key AS 'HostKey'
```

Mapping into InstallerEvidenceDetail_ci

Source alias	InstallerEvidenceDetail_ci property	Type	Max
Evidence	Evidence	nvarchar	64
HostKey	HostKey	varchar	64
InstallDate	InstallDate	nvarchar	255
Name	Name	nvarchar	512
ProductCode	ProductCode	nvarchar	110
Vendor	Vendor	nvarchar	400
Version	Version	nvarchar	144

LastLoggedOnUserDetail_ci (Staging Table)

Data in this staging table may also be saved in `LastLoggedOnUserDetail.xml` on the staging server.

BMC Discovery query

This query extracts data from the `Detail` node in the BMC Discovery datastore, and provides the aliases for properties referenced in the table below:

```
SEARCH Detail
   WHERE (NODECOUNT(TRAVERSE Detail:Detail:ElementWithDetail:Host) > 0)
    AND (type = 'FNMP_LastLoggedOnUser')
   SHOW name AS 'UserName',
    #Detail:Detail:ElementWithDetail:Host.key AS 'HostKey'
```

Mapping into LastLoggedOnUserDetail_ci

Source alias	LastLoggedOnUserDetail_ci property	Type	Max
HostKey	HostKey	varchar	64
UserName	UserName	nvarchar	255

NetworkInterface_ci (Staging Table)

Data in this staging table may also be saved in `NetworkInterface.xml` on the staging server.

BMC Discovery query

This query extracts data from the `NetworkInterface` node in the BMC Discovery datastore, and provides the aliases for properties referenced in the table below:

```
SEARCH NetworkInterface
   WHERE (NODECOUNT
            (TRAVERSE InterfaceOfDevice:DeviceInterface:
                      DeviceWithInterface:Host) > 0)
   SHOW
    (ip_addr
      OR (NODECOUNT(TRAVERSE InterfaceWithAddress:InterfaceAddress:
                      IPv4Address:IPAddress) > 0
       AND fmt("%.510s", join(#InterfaceWithAddress:InterfaceAddress:
                      IPv4Address:IPAddress.ip_addr, ', ')))
      OR ''
     ) AS 'IP',
    mac_addr AS 'MAC',
    #InterfaceOfDevice:DeviceInterface:DeviceWithInterface:Host.key
       AS 'HostKey'
```

Mapping into NetworkInterface_ci

Source alias	NetworkInterface_ci property	Type	Max
HostKey	HostKey	varchar	64
IP	IP	nvarchar	512
MAC	MAC	nvarchar	512

SoftwareInstance_ci (Staging Table)

Data in this staging table may also be saved in `SoftwareInstance.xml` on the staging server.

BMC Discovery query

This query extracts data from the `SoftwareInstance` node in the BMC Discovery datastore, and provides the aliases for properties referenced in the table below:

```
SEARCH SoftwareInstance
   WHERE (NODECOUNT(TRAVERSE RunningSoftware:HostedSoftware:Host:Host) > 0)
    AND vm_type IS NOT DEFINED
   SHOW
    type + (edition AND (' ' + edition) OR '') AS 'Name',
    product_version AS 'Version',
    (publisher OR single(#Element:Maintainer:Pattern:Pattern.publishers))
       AS 'Publisher',
    EXPLODE #RunningSoftware:HostedSoftware:Host:Host.key AS 'HostKey'
```

Mapping into SoftwareInstance_ci

Source alias	SoftwareInstance_ci property	Type	Max
HostKey	HostKey	varchar	64
Name	Name	nvarchar	255
Version	Version	nvarchar	255
Publisher	Publisher	nvarchar	255

SoftwareInstanceVirtualMachine_ci (Staging Table)

Data in this staging table may also be saved in `SoftwareInstanceVirtualMachine.xml` on the staging server.

BMC Discovery query

This query extracts data from the `SoftwareInstance` node in the BMC Discovery datastore, and provides the aliases for properties referenced in the table below (note that the `regex` is wrapped for printing, and should appear all on

one line):

```
SEARCH SoftwareInstance
   WHERE (NODECOUNT(TRAVERSE RunningSoftware:HostedSoftware:Host:Host) > 0)
    AND vm_type IS DEFINED
   SHOW
    vm_type AS 'VMType',
    (zone_uuid OR vm_bios_serial) AS 'VMSerial',
    (zonename OR vm_name) AS 'VMName',
    (zoneid OR
      (vm_type = 'VMware Virtual Machine'
       AND extract(vm_uuid, regex "^(..) (..) (..) (..) (..) (..) (..)
                                   (..)-(..) (..) (..) (..) (..) (..)
                                   (..) (..)$",
        regex "\1\2\3\4-\5\6-\7\8-\9\10-\11\12\13\14\15\16")
       OR vm_uuid
      )
     ) AS VMID,
    vm_guest_os AS 'GuestFullName',
    zonepath AS 'VMPath',
    formatTime(last_update_success, "%FT%T") as 'LastUpdateSuccess',
    single(#HostContainer:HostContainment:ContainedHost:Host.key)
       AS 'GuestHostKey',
    single(#RunningSoftware:HostedSoftware:Host:Host.key) AS 'HostKey'
```

Mapping into SoftwareInstanceVirtualMachine_ci

Source alias	SoftwareInstanceVirtualMachine_ci property	Type	Max
GuestFullName	GuestFullName	nvarchar	512
GuestHostKey	GuestHostKey	varchar	64
HostKey	HostKey	varchar	64
LastUpdateSuccess	LastUpdateSuccess	datetime	
VMID	VMID	nvarchar	255
VMName	VMName	nvarchar	512
VMPath	VMPath	nvarchar	512
VMSerial	VMSerial	nvarchar	255
VMType	VMType	nvarchar	255

Staging to FlexNet Manager Suite

This section provides the mapping from the staging database to the final presentation in the web interface for FlexNet

Manager Suite.

This section is organized alphabetically by object names in FlexNet Manager Suite (for example, 'computers' are known as 'inventory devices'). This is useful when you are starting from data displayed in FlexNet Manager Suite, tracing it back to the staging database, and then using the preceding section to see the source of that data point in BMC Discovery.

If you are coming the other direction, it is easiest to use your PDF search tool to find the staging database table name, a dot separator, and the property name. For example, to find out where the `Hostname` property from the `Host_ci` table of the staging database ends up, search for `Host_ci.Hostname`. You will find it listed against the **Name** property in the **All Inventory** page (and other similar listings) of the web interface .

For each object-based topic, the following details are included:

- Whether the import of new data that was not previously in the compliance database creates new records on import.
- Whether the import of changed data for records with matching key values is allowed to update the compliance database.
- Whether the absence of records previously imported from *only* the BMC Discovery source causes the old records to be removed from the compliance database.
- The source table and property (in dot-separated format) from the staging database. Look back to the previous section to check data types and maximum string sizes in the staging database.
- The property name in the destination page of the web interface. Each listing is sorted alphabetically on this property name. Where a property forms part of a compound key identifying records in the underlying compliance database, it is appropriately marked as a partial key ("p/key").
- The data type of the property.
- For string values, the maximum length allowed in the web interface (which is also normally the maximum string length available in compliance database). If this is less than the string length allowed in the staging database, the string is normally truncated to length (except as noted in the comments for an individual property).
- Any general notes.

Inventory Device (Computer)

Inventory devices are (in the main) computers that have been found in imported inventory; and in the current context, they are found in data imported from BMC Discovery. (Keep in mind that it is possible for a device to be reported in more than one inventory source, which becomes important when you are assessing automatic deletion of records no longer appearing in the source inventory.)

FlexNet Manager Suite correlates data from several points within BMC Discovery to create inventory device records. For example, the main sources from lowest to highest priority are:

1. The `HostInfo` node, by default populated during the discovery phrase of data gathering by BMC Discovery. Current data is staged in the `HostInfo_ci` staging table; but if BMC Discovery does not inventory the target device regularly, its information may be cleaned up and lost from this node.
2. The `Host` node, which by default BMC Discovery populates with data from the `HostInfo` node.
3. Data (of type `CPU Information`) visible in the `Detail` node of BMC Discovery, and staged into the

CPUInformationDetail_ci staging table.

Tip: *This data depends on a custom CPU pattern available only by request (see CPUInformationDetail_ci (Staging Table) for details).*

4. The optional UnixHardwareData pattern (delivered as part of the BMC Discovery adapter). Data from this source is also visible in the Detail node of BMC Discovery (with the type FNMP_HardwareEvidence), and is staged in the HardwareEvidenceDetail_ci table.

Generally, after import into FlexNet Manager Suite, inventory devices are stored in the ComplianceComputer table of the underlying compliance database.

Because inventory devices have so many properties, the following tables group their properties into smaller subsets. Within each subset, the tables are sorted alphabetically by the **Display property** column.

- Are new records created by imports from the staging database table? — **Yes**.
- Can records with matching compound keys be updated by these imports? — **Yes**.
- Are records unmatched by an import from the staging table deleted from the compliance database (provided that the staging database for BMC Discovery data is the last/only source for these records)? — **Yes**.

Inventory device: Computer (general properties)

All these properties are available in listings such as the **All Inventory** page; and within the properties of an individual inventory device, these all appear on the **General** tab.

Source table.property	Display property	Type	Max	Notes
Host_ci.Domain	**Domain name** (p/key)	String	100	
Host_ci.Serial	**Firmware serial number**	String	100	Visible in the **General** tab of the inventory device properties. Identifies the firmware on the inventory device.
NetworkInterface_ci.IP	**IP address**	String	256	
NetworkInterface_ci.MAC	**MAC address**	String	256	
Host_ci.Vendor	**Manufacturer**	String	128	
Host_ci.Model	**Model**	String	128	
Host_ci.Hostname	**Name** (p/key)	String	256	
Host.Serial	**Serial number**	String	100	A duplicate of Host_ci.Serial, this time taken from the Host database view.

Inventory device: Hardware properties

Except as noted, these properties are available in listings such as the **All Inventory** page; and within the properties of an individual inventory device, these mostly appear on the **Hardware** tab (where several values may be manually overridden if the collected inventory data is inaccurate).

Source table.property	Display property	Type	Max	Notes
`Host_ci.ProcessorSpeed`	**Clock speed (MHz)**	Integer		The maximum clock speed of the fastest processor in the computer in megahertz.
`Host_ci.Cores` (or `CPUInformationDetail_ci.Cores` for ADDM 11.0 or earlier); or, when the optional `UnixHardwareData` pattern has been used, `HardwareEvidenceDetail_ci.Cores`	**Cores**	Integer		The number of cores in the device (or, for a virtual machine, the number of cores assigned to the VM).
`FileSystem_ci.Size` (summed)	**Disk (GB)**	Integer		All `FileSystem_ci` records with an identical `HostKey` are selected from the staging database, and their `Size` values are summed and the result converted to GB for display.
Not applicable	**Display adapters**	Integer		The number of display adapters is not returned by BMC Discovery.
`FileSystem_ci.COUNT(HostKey)`	**Hard drives**	Integer		
Not applicable	**Inventory chassis type**	String	128	After import from BMC Discovery, this value is always `Unknown` (although some inventory tools can return the chassis type, BMC Discovery is not among them). The result is only available in the property sheet for an inventory device (and its linked asset, if any), and not in device listings. There is also an **Assigned chassis type** where you can correct inventory data if required.

Source table.property	Display property	Type	Max	Notes
`NetworkInterface_ci.IP` (counted)	**Network cards**	Integer		The number of network cards in the device is not returned by BMC Discovery. Instead, the count of distinct IP-enabled interfaces sharing a common `HostKey` is used.
`Host_ci.LogicalProcessors` (or `CPUInformationDetail_ci.LogicalProcessors` for ADDM 11.0 or earlier); or, when the optional `UnixHardwareData` pattern has been used, `HardwareEvidenceDetail_ci.Cores`	**Threads**	Integer		Saved in the `ComplianceComputer` table in the compliance database, and generally represented in the web interface as **Threads** for virtual machines.
`Host_ci.OS`	**Operating system**	String	128	
`Host_ci.Processors` (or `CPUInformationDetail_ci.Processors` (for ADDM 11.0 or earlier); or, when the optional `UnixHardwareData` pattern has been used, `HardwareEvidenceDetail_ci.Cores`	**Processors**	Integer		The number of processors reported in the inventory device.
`Host_ci.ProcessorType`	**Processor type**	String	256	
`Host_ci.RAM`	**RAM (GB)**	String	128	The imported value (in bytes) is converted to GB for display.
`Host_ci.ServicePack`	**Service pack**	String	128	
n/a	**Sockets**	Integer		The number of sockets — a value that is not reported by BMC Discovery.

Inventory device: Last inventory details

These properties are available in listings such as the **All Inventory** page; and within the properties of an individual inventory device, these appear on the **General** tab.

Source table.property	Display property	Type	Max	Notes
`HostInfo_ci.InventoryDate`	**Last inventory date**	Date		For virtual hosts, the most recent inventory date for any guest system may update this value.

Source table.property	Display property	Type	Max	Notes
`BMC Atrium Discovery and Dependency Mapping`	**Last inventory source**	String	128	The string literal "BMC Atrium Discovery and Dependency Mapping" is always inserted as the source value when using this adapter.

Inventory device: User details

These properties are available in listings such as the **All Inventory** page; and within the properties of an individual inventory device, these appear on the **Ownership** tab.

Source table.property	Display property	Type	Max	Notes
n/a	**Assigned user**	String	512	This value is set manually by an operator in FlexNet Manager Suite.
n/a	**Calculated user**	String	512	Set automatically as the logged-on user seen most frequently in the last 10 inventory imports.
`LastLoggedOnUserDetail_ci.Username` (when the optional pattern `WindowsLastLoggedOnUser` has been used)	**Last logged on user**	String	512	This information is available only for Windows platforms, and only when `WindowsLastLoggedOnUser` has been used. On data import, if the username cannot be matched in the `ComplianceUser` table of the compliance database, a new record is created, and appropriately linked to the `ComplianceComputer` record.

Installer Evidence

Installer evidence is a record of software installation on an inventory device. 'Normalized' installer evidence (meaning installer evidence that includes standard string values as noted below) allows the automatic matching of the evidence to the application that it represents, using the Application Recognition Library (ARL), frequently updated by Flexera.

- Are new records created by imports from the staging database table? — **Yes**.
- Can records with matching compound keys be updated by these imports? — **No**.
- Are records unmatched by an import from the staging table deleted from the compliance database (provided that the staging database for BMC Discovery data is the last/only source for these records)? — **Yes** (records linking applications to inventory devices are deleted under these conditions).

Installer evidence mapping

Most of these properties are available in listings such as the **Installer evidence** tab of the **All Evidence** page; and within the properties of an individual installer evidence record, these appear on the **General** tab.

Source table.property	Display property	Type	Max	Notes
`Host_ci.HostKey`	Not displayed. (p/key)	n/a		Identifies the inventory device on which the installer evidence was found. This value is not displayed in listings of, nor properties for, installer evidence (because in those contexts, the installer evidence may be relevant to many different devices). However, the link from the installer evidence to the application, plus this link to an individual device where the installer evidence has been found, allows the list of installed applications to appear on the **Applications** tab of the inventory device properties.
`DiscoveredPackages_ci.Name`	**Name** (p/key)	String	256	For recognition using the standard ARL to work reliably, this value must be the `Display Name` from `Add/Remove Programs` on Windows (or equivalent).
`DiscoveredPackages_ci.Publisher`	**Publisher** (p/key)	String	200	For recognition using the standard ARL to work reliably, this value must be the `Publisher` from `Add/Remove Programs` on Windows (or equivalent).
`ADDM` (string literal)	**Type**	String	32	The string literal "ADDM" is supplied as the installer evidence type for this adapter.
`DiscoveredPackages_ci.Version`	**Version** (p/key)	String	72	For recognition using the standard ARL to work reliably, this value must be the `Display Version` from `Add/Remove Programs` on Windows (or equivalent).

File Evidence

File evidence consists of files found on the disk of an inventory device. 'Normalized' file evidence (meaning file evidence that includes standard string values as noted below) allows the automatic matching of the file evidence to the application of which it forms part, using the Application Recognition Library (ARL), frequently updated by Flexera.

- Are new records created by imports from the staging database table? — **Yes**.
- Can records with matching compound keys be updated by these imports? — **No**.
- Are records unmatched by an import from the staging table deleted from the compliance database (provided that the staging database for BMC Discovery data is the last/only source for these records)? — **Yes**.

File evidence mapping

Most of these properties are available in listings such as the **File evidence** tab of the **All Evidence** page; and within the properties of an individual file evidence record, these appear on the **General** tab. (This listing is alphabetical by **Display property** values.)

Source table.property	Display property	Type	Max	Notes
`Host_ci.HostKey`	Not displayed. (p/key)	n/a		Identifies the inventory device on which the file evidence was found. This value is not displayed in listings of, nor properties for, file evidence (because in those contexts, the file evidence may be relevant to many different devices). However, once the file evidence is linked to an application, this link to an individual device where the file evidence has been found may affect the list of installed applications that appears on the **Applications** tab of the inventory device properties.
`FileEvidenceDetail_ci.Company`	**Company** (p/key)	String	100	For recognition using the standard ARL to work reliably, this value must be the `Company` WinPE header property from Windows executables.
`FileEvidenceDetail_ci.Description`	**Description** (p/key)	String	200	For recognition using the standard ARL to work reliably, this value must be the `Description` WinPE header property from Windows executables.
`FileEvidenceDetail_ci.FileName`	**Name** (in listings) or **File name** (in file evidence properties) (p/key)	String	256	For recognition using the standard ARL to work reliably, this value must be: • On Windows, the file name without path • On UNIX-like platforms, the full absolute file path.
`FileEvidenceDetail_ci.FileSize`	**Size**	Integer		
`FileEvidenceDetail_ci.FileVersion`	**Version** (p/key)	String	100	For recognition using the standard ARL to work reliably, this value must be the `File Version` WinPE header property from Windows executables.

The following extended properties of file evidence records are not populated in an BMC Discovery import:

- **File path**

- **Language**
- **Product name**
- **Product version**.

WMI Evidence

Windows Management Instrumentation (WMI) evidence is most often customized to help recognize operating systems, and (when the FlexNet inventory agent is in use) to help recognize aspects of special applications like Microsoft SQL Server. 'Normalized' WMI evidence (meaning WMI evidence that includes standard string values as noted below) allows the automatic matching of the WMI evidence to the application to which it relates, using the Application Recognition Library (ARL), frequently updated by Flexera.

- Are new records created by imports from the staging database table? — **Yes**.
- Can records with matching compound keys be updated by these imports? — **No**.
- Are records unmatched by an import from the staging table deleted from the compliance database (provided that the staging database for BMC Discovery data is the last/only source for these records)? — **Yes**.

WMI evidence mapping

In the web interface for FlexNet Manager Suite, there is no separate listing of WMI evidence. It is only possible to see these properties using the search facility in the **WMI** section of the **Evidence** tab of application properties. (This listing is alphabetical by **Display property** values.)

Source table.property	Display property	Type	Max	Notes
`Host_ci.HostKey`	Not displayed. (p/key)	n/a		Identifies the inventory device on which the WMI evidence was found. This value is not displayed in the application properties **Evidence** tab. However, once the WMI evidence is linked to an application, this link to an individual device where the WMI evidence has been found may affect the list of installed applications that appears on the **Applications** tab of the inventory device properties.
`Caption` (string literal)	**Property name** (p/key)	String	50	The string literal "Caption" is inserted for all WMI evidence records from BMC Discovery.
`Host_ci.OS` (left-most 256 characters)	**Property value** (p/key)	String	256	For OS recognition using the standard ARL to work reliably, this value must be the equivalent of the `Win32_OperatingSystem.Name` WMI property.

Source table.property	Display property	Type	Max	Notes
`Host_ci.OSType` converted as per notes.	**WMI class** (p/key)	String	50	The BMC Discovery data converts to standard class types used by FlexNet Manager Suite as follows: • `Windows` becomes `Win32_OperatingSystem` • Either of `VMware ESX` or `VMware ESXi` becomes `VMWARE_OperatingSystem` • Any other value is replaced by `MGS_OperatingSystem`.

App-V Server Adapter

Microsoft App-V (full name Microsoft Application Virtualization) is an application virtualization and application streaming solution. It allows access to applications in three different ways:

- Users may stream applications directly from a central App-V Management Server to their client computers, executing the code locally in light virtual machines that provide a protective 'bubble' around the executing software.
- The applications may be deployed using Microsoft System Center Configuration Manager (SCCM).
- Applications may be deployed in 'stand alone' mode, such as manual delivery through file shares or on a USB stick, without the use of any server infrastructure.

Applications delivered in any of these ways require licensing, and you can manage the appropriate licenses using FlexNet Manager Suite:

- Applications streamed from the App-V Publishing Server(s) are monitored using the App-V server adapter supplied as a standard part of FlexNet Manager Suite, and (for App-V release 5.0 and later) the `AppVMgmtSvr.ps1` PowerShell script installed on the App-V Management Server. Both the App-V server adapter and the `AppVMgmtSvr.ps1` PowerShell script are documented in this chapter.
- Applications deployed using Microsoft SCCM are recorded automatically as part of the standard inventory import from SCCM.
- Applications deployed manually can be recorded manually in FlexNet Manager Suite. Manual deployment is not a recommended best practice because of the inherent difficulties in management and demonstrating compliance, and there is no automation possible through FlexNet direct inventory gathering to cover manual deployment.

Supported versions

The App-V server adapter supports the current version of FlexNet Manager Suite, and releases 4.6, 5.0, and 5.1 of Microsoft Application Virtualization.

Because of significant architectural change between release 4.6 and release 5.0 of App-V, there are matching significant differences in the App-V server adapter for the different versions. It is important to read this document carefully, noting the differences that apply to "release 4.6" and "release 5.0 and later".

1

Architecture, Components, and Prerequisites

Following the changes that Microsoft made in the architecture of App-V between release 4.6 and release 5.0, the App-V server adapter is also significantly different when interfacing to the different App-V releases. There are separate chapters for each different architecture. Identify the release of Microsoft App-V in use in your enterprise, and focus on the architecture that is appropriate to that release.

Architecture and Operation for App-V 4.6

This discussion applies to use of Microsoft App-V server infrastructure, streaming applications to App-V clients on end-point devices. (Where applications are instead installed by Microsoft SCCM, use the inventory import from SCCM instead of this adapter.)

In its streaming implementation, Microsoft App-V release 4.6 has three main kinds of components:

- A database (referred to here as the App-V Management Server database), which may be on a separate server
- One or more Management Servers that access the App-V Management Server database and provide a user interface for system control
- One or more streaming servers that may directly deliver application packages.

Of these, only the App-V Management Server database is relevant to the App-V server adapter for FlexNet Manager Suite.

Prerequisites

Operation requires that you have:

- A supported version of Microsoft App-V (see App-V Server Adapter).
- An operational App-V Management Server database.
- A FlexNet inventory beacon that has network access to your App-V Management Server database, and is also able to upload gathered inventory to the central FlexNet Manager Suite server (either directly or through a hierarchy of inventory beacons).

- An inventory beacon importing Active Directory data from the same domain where the App-V server resides. (This may be the *same* inventory beacon that runs the App-V server adapter, but this is not a requirement.)

  Tip: *If you have App-V applications secured by security groups from multiple Active Directory domains, ensure that the Active Directory import runs against all applicable domains in your environment. The simplest approach may well be to ensure that you import from all your Active Directory domains, since if you use foreign security principals from multiple trusted domains, it can be difficult to keep track of access to App-V packages. FlexNet Manager Suite imports only from each individually specified Active Directory domain; so you need to ensure that all applicable domains are specified. As an example of multiple domains being affected:*

  - *Suppose you have Group-A in Domain-A that contains a child Group-B, where Group-B actually comes from Domain-B.*
  - *In this case, granting access to an App-V package to Group-A also grants access to Group-B (because of the parent-child relationship between the groups).*
  - *This inheritance continues to work even when there is only one-way trust from Domain-B to Domain-A.*
  - *In such a case, it is imperative that you run an Active Directory import against both Domain-A and Domain-B. When you have many domains, the simplest path is just to run an Active Directory import from every domain.*

- Operators who can identify the applications represented by the App-V packages, and link those applications to the appropriate licenses.

Tip: *You may have multiple App-V Management Servers, and multiple streaming servers, that link to a single App-V Management Server database. This requires only one connection from the FlexNet Manager Suite App-V server adapter, because this connects only to the database. However, if you have multiple App-V Management Server databases in your estate, configure a separate connection to each of them on appropriate inventory beacons. Where helpful, you may configure multiple such connections (each separately scheduled as you choose) on one inventory beacon.*

In operation

The following diagram shows the operational architecture for the App-V server adapter for App-V release 4.6.

The numbers here refer to the numbers shown in the diagram above:

1. The inventory beacon imports data from Active Directory, including groups (and their members), users, and computers, and the security identifiers for each item within Active Directory. (These security identifiers, or SIDs, are the same identifiers that App-V reports for usage of the applications delivered through App-V packages.)
   - These are immediately uploaded to the central application server for FlexNet Manager Suite.
   - As soon as the upload is completed, the data is imported into the compliance database.
2. On the schedule you specify on the inventory beacon, the App-V adapter:
   - Connects to the App-V Management Server database
   - Imports a list of the App-V packages from the database, and the access control lists (ACLs) that determine which Active Directory groups and users have access to the applications inside the packages. The latter are identified by their security identifiers (SIDs).
   - Immediately uploads the data to the central application server for FlexNet Manager Suite. (If the upload fails for some reason, there is a catch-up upload task that by default is scheduled overnight.)
   - The data waits in the staging area on the central application server for the next scheduled inventory import and compliance calculation (by default, scheduled overnight).
3. When information about a new App-V package is first imported, an operator must identify the package and link it (like installer evidence) to an application record. This work must be done manually because (in release 4.6) App-V packages are opaque about the applications they contain. As well, for any meaningful calculations of consumption, the application must be linked to a suitable license. This linking effort is required only for the first import of each new package.

Once the links are established, each subsequent compliance calculation assigns consumption by the correct users and computers to the appropriate (linked) license. This consumption information is then available both in the management views and in reports.

Architecture and Operation for App-V 5 and Later

This discussion applies to use of Microsoft App-V server infrastructure, streaming applications to App-V clients on end-point devices. (Where applications are instead installed by Microsoft SCCM, use the inventory import from SCCM instead of this adapter.)

In its streaming implementation, Microsoft App-V release 5.0 (and later) has the following components, apart from the App-V clients:

- A database (referred to here as the App-V Management Server database), which may be on a separate server
- A separate reporting database (referred to here as the App-V reporting database), which may also be on a separate server (importantly, this database stores application usage information)
- One or more Management Servers that access the App-V Management Server database and provide a user interface for system control
- One or more Reporting Servers that access the App-V reporting database and provide operational reports to help

manage the App-V infrastructure

- One or more streaming servers (called App-V Publishing Servers) that may directly deliver application packages.

Of these, for App-V 5.0 and later, only the App-V reporting database and an App-V Management Server are relevant to the App-V server adapter for FlexNet Manager Suite. (If you are familiar with the adapter for release 4.6 of App-V, notice that we have switched databases, and added the Management Server — the architecture is completely different.)

Prerequisites

Operation requires that you have:

- A supported version of Microsoft App-V (see App-V Server Adapter).
- An operational App-V reporting database.
- An operational AppV Management Server.
- The `AppVMgmtSvr.ps1` PowerShell script installed, configured and scheduled on your AppV Management Server (see Obtaining (and Deploying) the Adapter Components for details). This is one of the significant changes since the previous adapter.
- A FlexNet inventory beacon that has network access to your App-V reporting database, and is also able to upload gathered inventory to the central FlexNet Manager Suite server (either directly or through a hierarchy of inventory beacons).
- An inventory beacon importing Active Directory data from the same domain where the App-V server resides. (This may be the *same* inventory beacon that runs the App-V server adapter, but this is not a requirement.)

  Tip: *If you have App-V applications secured by security groups from multiple Active Directory domains, ensure that the Active Directory import runs against all applicable domains in your environment. The simplest approach may well be to ensure that you import from all your Active Directory domains, since if you use foreign security principals from multiple trusted domains, it can be difficult to keep track of access to App-V packages. FlexNet Manager Suite imports only from each individually specified Active Directory domain; so you need to ensure that all applicable domains are specified. As an example of multiple domains being affected:*

  - *Suppose you have Group-A in Domain-A that contains a child Group-B, where Group-B actually comes from Domain-B.*
  - *In this case, granting access to an App-V package to Group-A also grants access to Group-B (because of the parent-child relationship between the groups).*
  - *This inheritance continues to work even when there is only one-way trust from Domain-B to Domain-A.*
  - *In such a case, it is imperative that you run an Active Directory import against both Domain-A and Domain-B. When you have many domains, the simplest path is just to run an Active Directory import from every domain.*

- Operators who can link the applications identified in the App-V packages to the appropriate licenses.

Tip: *You need only one connection from the FlexNet Manager Suite App-V server adapter (on an inventory beacon) to the App-V reporting database. This single App-V reporting database may support multiple App-V Management Servers, and multiple Publishing Servers; but only a single connection to the database is required.*

Limitation

For App-V release 5.0 and later, the system supports installation of the `AppVMgmtSvr.ps1` PowerShell script on only one App-V Management Server. This single Management Server may support multiple Publishing Servers (if necessary spread worldwide for faster distribution of App-V packages to App-V clients); and the App-V clients may report to multiple Reporting Servers (independent of the source from which the App-V packages were downloaded). Different App-V Management Servers do not self-identify in the `.raa` inventory file, and the App-V reporting database does not identify which application usage information is associated with which App-V Management Server. For these reasons, only a single App-V Management Server (for release 5.0 and later) is supported.

If your App-V (release 5.0 or later) environment has multiple Management Servers, choose one as the data source for App-V packages and the applications they contain. For example, if you have Production, Dev, and Test servers, place the `AppVMgmtSvr.ps1` PowerShell script on the Production App-V Management Server. Also ensure that the App-V server adapter (on an inventory beacon) connects to the matching Production App-V reporting database.

In operation

The following diagram shows the operational architecture for the App-V server adapter for release 5.0 and later.

The numbers here refer to the numbers shown in the diagram above:

1. The inventory beacon imports data from Active Directory, including groups (and their members), users, and computers, and the security identifiers for each item within Active Directory. (These security identifiers, or SIDs, are the same identifiers that App-V reports for usage of the applications delivered through App-V packages.)
   - These are immediately uploaded to the central application server for FlexNet Manager Suite.
   - As soon as the upload is completed, the data is imported into the compliance database.
2. On the schedule you specify on the App-V Management Server, the `AppVMgmtSvr.ps1` PowerShell script:
   - Uses the API to gather a list of the available App-V packages
   - Imports from the database, and the access control lists (ACLs) that determine which Active Directory groups and users have access to the applications inside the packages. The latter are identified by their security identifiers (SIDs)

- Uploads the collected data in a remote application access (`.raa`) file to its configured inventory beacon, which in turn uploads the file to the central application server for FlexNet Manager Suite.
- The data waits in the staging area on the central application server for the next scheduled inventory import and compliance calculation (by default, scheduled overnight).

3. On the schedule you specify on the inventory beacon, the App-V adapter:
   - Connects to the App-V reporting database
   - Imports App-V package usage by users and computers. These are all identified by their security identifiers (SIDs).
   - Immediately uploads the data to the central application server for FlexNet Manager Suite. (If the upload fails for some reason, there is a catch-up upload task that by default is scheduled overnight.)
   - The `.raa` file collected by the PowerShell script is uploaded and immediately resolved into staging tables in the database.

     Tip: *If you manually copy an `.raa` file to your application server, you can import it with the following command:*

     ```
     > mgsimport -t remoteApplication
     ```

   - The data waits in the staging area on the central application server for the next scheduled inventory import and compliance calculation (by default, scheduled overnight).
4. When the compliance calculation is run, FlexNet Manager Suite uses the uploaded SIDs to correlate the various data elements:
   - App-V packages are shown as installer evidence (based on the MSI information uploaded by the `AppVMgmtSvr.ps1` PowerShell script).
   - If an appropriate application record exists (either in the Application Recognition Library or as a locally-created record) with a suitable installer evidence rule, the installed evidence (package) is automatically matched with the application.
   - All users with access to an App-V package are shown as having an installation of the related application on every computer for which the user is either the assigned or calculated user.
   - All computers with access to an App-V package are shown as having an installation of the related application.
   - If the application is linked to a license, consumption is shown for the correct users and computers on that license (or, if it is linked to multiple licenses, on the highest priority license still having unconsumed entitlements). This consumption information is then available both in the management views and in reports. (If this is the first import to reveal an application in an App-V package, an operator needs to link the application record to an appropriate license.)

2

Set Up and Operations

This chapter covers the configuration of the adapter, and the work needed to enable application recognition from the imported inventory.

Keep clearly in mind the distinctions required for App-V release 4.6, and release 5.0 and later. For example, Obtaining (and Deploying) the Adapter Components documents processes needed only for release 5.0 and later; and Configuring the Adapter requires a distinct database connection in each of the cases.

Obtaining (and Deploying) the Adapter Components

1. Download the `Adapter Tools for` FlexNet Manager Suite `2019 R2.zip` archive from the Flexera Customer Community knowledge base:

   a. Access https://flexeracommunity.force.com/customer/articles/en_US/INFO/Adapter-Tools-for-FlexNet-Manager-Suite.

   Tip: *Access requires your Customer Community user name and password. If you do not have one, use the link on the login page to request one.*

   b. Click the link `Adapter Tools for FlexNet Manager Suite`.

   A new browser tab may appear temporarily, and the download of `Adapter Tools for` FlexNet Manager Suite `2019 R2.zip` commences.

   c. In your browser dialog, choose to save the file, and if the browser allows it, direct the saved file to a convenient working location (such as `C:\temp` on a central, accessible server).

   If your browser saves the file to a default location (such as your `Downloads` folder), move or copy it to the appropriate working location when the download is finished.

2. Right-click the zip archive, and choose **Extract All...**.

3. In the extracted archive, navigate to `Adapter Tools\App-V Management Server Agent\AppVManagementServer5`.

> ***Tip:*** *As an alternative to downloading the archive again, if you still have access to the 'CD image' you downloaded to install FlexNet Manager Suite itself on your application server(s), you can also find these files in the unzipped installation archive under* `FlexNet Manager Suite\Installers\App-V Management Server Agent\AppVManagementServer5`*.*

4. Use your preferred method to deploy the `AppVMgmtSvr.ps1` PowerShell script to your App-V Management Server.

   You may install the script in your preferred folder.

5. Use your preferred task scheduling technology to schedule data collection by the PowerShell script.

   Typically you want the `.raa` file uploaded to the central FlexNet Manager Suite operations databases before the system import and license calculations take place. By default, this occurs daily at 2am central server time. As a two-hour upload buffer should be more than adequate, this suggests (within a single time zone) a data collection trigger at around midnight.

   These example steps are for Windows Server 2012. Adjust as necessary for your server operating system, or your chosen scheduling tool.

   a. In Windows Explorer, navigate to **Control Panel** > **System and Security** > **Administrative Tools**, and double-click **Task Scheduler**.

      The **Task Scheduler** window appears.

   b. In the navigation tree on the left, select **Task Scheduler Library**, and then in the **Actions** list on the right, click **New Folder...**.

      A dialog appears for entering the folder name.

      A suggested value is `FlexNet Manager Suite`.

   c. Click **OK**, and select the new folder in the navigation tree.

   d. Select **Action** > **Create Task...**.

      The **Create Task** dialog appears.

   e. Enter an appropriate **Name**, such as `FlexNet Manager Agent for App-V 5.x`, and add any **Description** to help future maintenance of this task.

      Your description may be something like `Collects App-V data from the Management Server and uploads to an inventory beacon`.

f. Click **Change User or Group...**.

The **Select User, Service Account, or Group** dialog appears.

g. Enter the account name that is to run the scheduled task, and click **OK**.

An appropriate account:

- Can run a Windows scheduled task on the App-V Management Server
- Can execute the PowerShell script
- Is an App-V Management Server administrator
- May conveniently be a domain account that can upload the results (using HTTP PUT) to the inventory beacon, although a separate account and password can be configured in the command line (required only where that inventory beacon is using Basic Authentication — if the inventory beacon uses anonymous authentication, ignore this requirement).

h. Further down in the **Security options** group, select **Run whether user is logged in or not**.

i. Switch to the **Triggers** tab, and click **New...**.

The **New Trigger** dialog appears.

j. Ensure that the default setting **Begin the task** `On a schedule` is selected, set the parameters for the schedule, and from the **Advanced settings** group, be sure that **Enabled** is selected.

The suggested schedule is daily at or before midnight local time, but be sure that this suits the upload procedures for your enterprise.

k. Switch to the **Action** tab, and click **New...**.

The **New Action** dialog appears.

l. Ensure that the default **Action**, `Start a program`, is selected, and browse to your local copy of `AppVMgmtSvr.ps1`.

m. In the **Add arguments (optional)** field, specify all the command-line arguments you need for the agent.

All command line arguments are documented in Command Line for PowerShell Script. For common implementations, you need to define only the URL to the inventory beacon.

This example uploads the `.raa` file to the `flexnetbeacon` inventory beacon, using the credentials of the account running the scheduled task.

```
.\AppVMgmtSvr.ps1 -beaconUrl http://flexnetbeacon
```

Tip: *You need to specify only the basis server in the URL, Internally, the FlexNet Beacon service briefly saves the uploaded file to `%CommonAppData%\Flexera Software\Incoming\RemoteApplications` before uploading the file to its parent. (If security within your enterprise prevents the PowerShell script uploading the `.raa` file to a inventory beacon, you can arrange an alternative method to save the `.raa` file to this folder on a convenient inventory beacon, and it is automatically uploaded and processed from this point.)*

n. Click **OK**.

o. Optionally, make any preferred adjustments to the **Conditions** or **Settings** tabs (normally the defaults are acceptable).

p. Click **OK** to close the **Create Task** dialog.

   The new task appears in the list of scheduled tasks for this server.

q. Right-click the new task, and click **Run** in the context menu.

   This checks that the scheduled task completes successfully.

r. Validate operations in the following ways:

   - Review the log file (by default, `AppVMgmtSvr.log` in the same folder as the PowerShell script) for any errors or warning messages.
   - Review the content of the output file (by default, `FNMS_AppV.raa` in the same folder as the PowerShell script). This file contains the results of the most recent execution of the PowerShell script, and is replaced at each invocation of the script. For more details of the file format, see File Format for .raa.
   - If uploaded to the inventory beacon, check for the presence of the `FNMS_AppV.raa` output file in *`%CommonAppData%`*`\Flexera Software\Incoming\RemoteApplications` on the inventory beacon. Remember that you have only a brief time window to check this before it is uploaded to the central application server and removed (by default, around 10 minutes).

The `AppVMgmtSvr.ps1` PowerShell script is now configured on your App-V release 5.0 or later Management Server. You can now configure the adapter itself that runs on the inventory beacon (see Configuring the Adapter).

Command Line for PowerShell Script

The `AppVMgmtSvr.ps1` PowerShell script is required only when importing inventory from Microsoft App-V release 5.0 or later. (It is not required if you are using App-V release 4.6.)

For applicable releases, `AppVMgmtSvr.ps1` is installed on the App-V Management Server, where, on a schedule that you determine, it collects details of the available App-V packages, and the users and computers that have access to the packages. (Separately, usage information is collected by the App-V server adapter, with its separate configuration described in Configuring the Adapter.)

The following options are supported, both for running `AppVMgmtSvr.ps1` manually and for executing it from a scheduled task or other scheduling tool. None of the options is mandatory, although some are required for normal operation.

Syntax

Syntax:
`AppVMgmtSvr.ps1` [*options*...]

Options

`-beaconUrl` *`validURL`*

```
-logFilePath log_file
-outputFilePath output_file
-password password
-upload $true | $false
-username account
```

where

`-beaconUrl validURL`	The URL to the appropriate inventory beacon to which the script should upload the generated `.raa` file. Include the protocol (`HTTP` or `HTTPS`), and if your inventory beacon uses a non-default port, include the port number in the standard way.
	
	Note: *Include only the basic URL of the server. No internal paths are needed, as these details are added automatically by `AppVMgmtSvr.ps1`.*
	
	There is no default value for `beaconUrl`, so that in production use (when `-upload $true`), a value must be supplied. It can be omitted for local testing on the server when uploading is not required. If `-upload $true` and `beaconUrl` is not set, obviously the upload must fail.
	Example values:
	`http://flexnetbeacon.example.com`
	`https://flexnetbeacon.example.com:499`

`-logFilePath log_file`	A file path (either absolute, or relative to the folder in which the script is executing) and file name for the log file generated by `AppVMgmtSvr.ps1`. Enclose the path in double quotation marks. A file with consistent name and path over time is replaced at each execution, containing only the results of the last execution, thus preventing unmanaged storage requirements. When this option is not specified, the default value is `AppVMgmtSvr.log`, saved in the folder where the script is executing.
`-outputFilePath output_file`	A file path (either absolute, or relative to the folder in which the script is executing) and file name for the output remote application access (`.raa`) file generated by `AppVMgmtSvr.ps1`. Enclose the path in double quotation marks. If the same name and path is used, the file is overwritten at each run of the script. When the option is not specified, the default value is `FNMS_AppV.raa`, saved in the folder where the script is executing.
`-password "password"`	The password (in plain text) for the account specified in `username`. Omitted when that option is not required. If specified, enclose the value in double quotation marks. When required and not specified, the script uses details of the account running the process.
`-upload $true | $false`	A Boolean that determines whether to attempt uploading the output file to an inventory beacon identified in `beaconUrl`. When not specified, the default is true, requiring that `beaconUrl` is specified so that the upload can succeed.

`-username "account"`	The account name used to upload the generated `.raa` file to the inventory beacon identified in `beaconUrl`. This is not required for inventory beacons using anonymous authentication. It may be specified for inventory beacons using Windows Basic Authentication. If specified, enclose the value in double quotation marks. When it is specified, the matching password must be provided in the `-password` option. When required and not specified, the script uses details of the account running the process.

Examples

(Examples here may be line-wrapped for convenient presentation; but should be entered on a single command line.)

Collect the inventory from the App-V Management Server, saving the file locally for inspection:

```
.\AppVMgmtSvr.ps1 -upload $false
```

Similarly, collect the inventory, saving the output and logs to temporary locations to avoid overwriting the normal output:

```
.\AppVMgmtSvr.ps1 -outputFilePath "C:\temp\FNMS_AppV.raa"
                  -logFilePath "C:\temp\AppVMgmtSvr.log"
                  -upload $false
```

Collect the inventory, and upload it to the specified inventory beacon, using the username and password for the account currently running the script:

```
.\AppVMgmtSvr.ps1 -beaconUrl http://flexnetbeacon.example.com
```

Upload the collected inventory to the specified inventory beacon, using the `testdomain\administrator` account:

```
.\AppVMgmtSvr.ps1 -beaconUrl http://flexnetbeacon.example.com
                  -username "testdomain\administrator"
                  -password "somepassword"
```

File Format for .raa

The `AppVMgmtSvr.ps1` PowerShell script interrogates the App-V Management Server, and saves the resulting data in a remote application access file (filename extension `.raa`), by default called `FNMS_AppV.raa`. This is an XML file that encapsulates data about each application's App-V package, and the MSI installer information known to App-V.

Here is an excerpt from an `FNMS_AppV.raa` file (here line-wrapped for presentation):

```
<remoteApplications accessModeID="2">
  <app farmName=""
       appID="9b09bc0d-9634-4838-b0ba-8c256ef4710d"
       appName="Blender"
       appFileName=""
       appFileVersion=""
       appFilePublisher=""
```

```
      appFileDesc=""
      userSid=""
      serverName=""
      serverDomainName=""
      isStreamingProfile="1" />
  <msiData farmName=""
          appID="9b09bc0d-9634-4838-b0ba-8c256ef4710d"
          msiDisplayName="Blender"
          msiPublisher="Blender"
          msiVersion="1.0"
          msiProductCode="{DFFFE0C6-3E9A-44E9-9EC1-B5C92DCEE4AF}" />
  <app farmName=""
      appID="5d80bef6-de4a-44f6-b4f8-9aa6a657880e"
      appName="FileZilla_3.2.4.1_win32-setup"
      appFileName=""
      appFileVersion=""
      appFilePublisher=""
      appFileDesc=""
      userSid="S-1-5-21-1336908958-3350896562-3141117955-1690"
      serverName=""
      serverDomainName=""
      isStreamingProfile="1" />
  <msiData farmName=""
          appID="5d80bef6-de4a-44f6-b4f8-9aa6a657880e"
          msiDisplayName="FileZilla Client 3.2.4.1"
          msiPublisher=""
          msiVersion="3.2.4.1"
          msiProductCode="filezilla client" />
 <app farmName=""
      appID="02787044-d434-4e7d-8770-cda46c988de8"
      appName="AdobeReader9"
      appFileName=""
      appFileVersion=""
      appFilePublisher=""
      appFileDesc=""
      userSid="S-1-5-21-1336908958-3350896562-3141117955-1690"
      serverName=""
      serverDomainName=""
      isStreamingProfile="1" />
  <msiData farmName=""
          appID="02787044-d434-4e7d-8770-cda46c988de8"
          msiDisplayName="Adobe Reader 9.1"
          msiPublisher="Adobe Systems Incorporated"
          msiVersion="9.1.0"
          msiProductCode="{ac76ba86-7ad7-1033-7b44-a91000000001}" />
  ...
</remoteApplications>
```

Some points of interest to note:

- The complete set of MSI attributes needed for FlexNet Manager Suite makes use of Asset Intelligence on the App-V release 5.0+ system, which saves a series of `AssetIntelligenceProperties` in the manifest file. From these properties, `AppVMgmtSvr.ps1` extracts the four `msi...` properties shown for the applications above.
- Not all applications deployed through App-V use MSI, as some applications use third-party installers. App-V packages using third-party installers cannot include the Asset Intelligence properties available through MSI. For such cases, the `AppVMgmtSvr.ps1` PowerShell script interrogates the App-V application registry to get complete installer evidence, populating the same attributes in the `.raa` file. (Thus the presence of `AssetIntelligenceProperties` in the manifest file does not necessarily mean that Asset Intelligence was available; but does mean that equivalent data has been obtained.)
- The `msiDisplayName` becomes the **Name** property of the installer evidence record, the `msiPublisher` maps to **Publisher**, `msiVersion` maps to **Version**. As with all installer evidence, the resulting record is matched against installer evidence "rules" (previously-recorded installer evidence records, most often generalized with judicious use of wild card `%` characters), and when matched, adds an installation count to the application linked to the rule. This means that in listings of installer evidence, the visible entry remains the generalized rule that matched our individual piece of installer evidence. However, the individual installer evidence record is visible by drilling down into the properties of the inventory device on which the inventory evidence was found.
- For a worked example of how the installer evidence for the FileZilla application in the extract above is matched against an installer evidence rule, see Investigating Issues.

Configuring the Adapter

The (core) App-V server adapter is set up on an inventory beacon. Only two tasks are required for configuring this built-in adapter:

- Specifying the connection to the appropriate database. There are distinct databases in the two versions:
  - For App-V release 4.6, the App-V server adapter connects to the Microsoft App-V Management Server database
  - For App-V release 5.0 and later, the App-V server adapter connects to the Microsoft App-V reporting database.
- Scheduling the imports.

Both tasks are summarized here. Further details are available in the inventory beacon help.

To configure the App-V server adapter (summary):

1. On the appropriate inventory beacon, start the FlexNet Beacon interface.

   An appropriate inventory beacon has network access to the appropriate database as described above; and it can upload to the central FlexNet Manager Suite server, either directly or through a hierarchy of inventory beacons.

   Tip: *Remember that logging into an inventory beacon requires an account with administrator privileges.*

2. Select the **Inventory systems** page in the FlexNet Beacon interface.
3. At the bottom of the page, click **New...**.

   The **Create SQL Source Connection** dialog opens.

4. Complete the values required in this dialog:

Connection name	A descriptive name for this connection that you will recognize later in lists.
Source Type	Select **App-V Standalone**. (Use this same value whether you are connecting to App-V release 4.6, or release 5.0 or later.) ***Tip:*** *'Standalone' means that you are using the adapter to connect directly to the appropriate App-V database, rather than collecting inventory through another source such as Microsoft SCCM.*
Server	Type the server name or IP address. If the database instance you need is not the default one on the server you identify, add the instance name, separated with a backslash character.
Authentication	Select one of: • **Windows Authentication** — Select this option to use standard Windows authentication to access the database server. The credentials of the account (on the inventory beacon) running the scheduled task for importing inventory are used to access the SQL Server database. This account must be added to an Active Directory security group that has access to the database. • **Windows (specific account)** — Use the following two fields (enabled when you make this choice) to specify an account on the inventory beacon that can make a connection to the SQL database. • **SQL Authentication** — Use the following two fields to specify an account and password registered as a user with database access on SQL Server . This account is used to access the database, regardless of the local account running the scheduled task on the inventory beacon server. ***Tip:*** *The account used needs read-only privileges.*
Username	The account name used for **SQL authentication**, or **Windows (specific account)**. (Not required for **Windows Authentication**.)
Password	The password for the account name required for **SQL authentication**, or **Windows (specific account)**. (Not required for **Windows Authentication**.)
Database	Enter the name of the database, or use the pull-down list to select from database names automatically detected on your specified server.

Connection is in test mode (do not import results)	Ensure that this check box is clear for production use. (For more details, see *Managing Microsoft SQL Server Database Connections* in the online help for FlexNet Manager Suite, in the section covering the inventory beacon.) ***Tip:*** *When using App-V release 4.6, you cannot complete configuration for operation of this adapter (specifically, you cannot map the App-V packages to real applications) until you have run an import in production mode, with this check box clear.*

Overlapping Inventory Filter	If you use more than one inventory source, it is possible to get overlapping inventory (records about the same endpoint device from multiple inventory tools). Because of differences between inventory tools, the overlapping inventory records may contain slightly different data. In the compliance browser (in FlexNet Manager Suite), you may nominate one inventory source as **Primary**, which gives its collected data priority for hardware properties imported from target inventory devices. The choices here give another, separate axis of control, based on whether or not the data from this particular source is to be considered "stale". This is different from the **Primary** setting in the following ways:
	• It is assessed *first* (before the primary setting is taken into account), and so may even modify the effect of your **Primary** setting. It is best practice to make sure that your chosen primary inventory source is *not* marked as stale with the following settings.
	• It affects not only imported *hardware* properties, but also *software* inventory (installer evidence, file evidence, and so on), as follows:
	  ◦ With two inventory sources (when neither one is considered stale), the total software inventory is a union (merging) of the results from both sources. This is useful when two different inventory tools have different specializations for software detection: the union means you are not blind-sided by a software inventory tool that missed something you should have licensed. This is considered the 'normal' operational case in a stable environment using multiple inventory sources.
	  ◦ When one of the two available sources is declared "stale" (using either of the first two choices below), all of its overlapping software inventory is excluded from the possible union of data sources (hardware inventory too, but here we're considering software). This behavior is valuable, for instance, when a target inventory device has migrated from one inventory source to another (perhaps by moving offices), but has not yet been obsoleted from this first source. Imagine that, as part of the office move, the MyApp application had also been uninstalled from the inventory device. You do not want the old and stale record from this source insisting there's an installation of MyApp when in fact it's no longer on the device. Of course, best practice is to obsolete the device from this stale source so that it is no longer reported in this inventory source; but these first two settings allow a quick control of *all* the overlapping data from this source, rather than having to obsolete devices one by one in the source tool. Another common scenario is when you are migrating over time from an old version of your chosen inventory tool to a newer version (on a new connection), and during transition both systems are still running. Declaring the old version's connection as "stale" means that as soon as a record appears in the new version's inventory for a particular device, the old device import is automatically superseded, and updates rely entirely on the new version of your inventory tool.
	Each of the following settings only takes affect when FlexNet Manager Suite is comparing the inventory dates of a device from this current inventory source and an

	overlapping record from another inventory source: • **Ignore the device's inventory from this data source** — When you have *more recent* inventory from another source for the same target inventory device, the record from this source is completely ignored. (Technically, the device record is deleted from the staging table in the database, and so can never be imported.) • **Ignore this device's inventory if older than *nn* days** — If you select this option, overlapping inventory collected by this source more than the set number of days before the import is ignored. Fresher overlapping data is still imported and considered for data merging. ***Tip:*** *In the interests of keeping inventory current, this control has a maximum value of 60 days.* • **Import the inventory from this source for possible merging** — Choose this option (the default) to declare that overlapping inventory collected from this connection is never considered stale. This is the normal operating setting when you are not trying to manage transitions from one inventory source to another. With this setting, overlapping records are merged in this way: **a.** If a data point exists in the **Primary** inventory source, it is used **b.** If two equal-priority sources have different inventory dates, the data point is taken from the most recent inventory **c.** As a tie-breaker, the connection ID for this source recorded in the database is used (normally meaning that the earliest-created inventory source has priority).

5. Click **Test Connection**.

   - If the inventory beacon can successfully connect to the nominated database using the details supplied, a `Database connection succeeded` message displays. Click **OK** to close the message. Click **Save** to complete the addition. The connection is added to (or updated in) the list.
   - If the inventory beacon cannot connect, a `Database connection failed` message is displayed, with information about why that connection could not be made. Click **OK** to close the message. Edit the connection details and retest the connection.

   You cannot save the connection details if the connection test fails. If you cannot get the connection test to succeed, click **Cancel** to cancel the addition of these connection details.

6. If you do not already have a schedule specified that can be used to run the adapter for this connection, create one now (see *Creating a Data Gathering Schedule* in the online help).

7. With the connection for this adapter selected in the **Inventory systems** page, click **Schedule...**.

   The **Select Schedule** dialog opens.

8. From the drop-down list, select the schedule you wish to apply.

Tip: *As you select each schedule from the list, the area below displays a summary of the schedule settings and the expected* ***Next run time*** *for this schedule.*

9. Click **OK** to apply the selected schedule.
10. Click **Save** to store these details.

    The list of connections is updated, and the **Next run** column for your selected connection shows the projected run time from the schedule you just attached.

11. With the connection for this adapter still selected in the **Inventory systems** page, click **Execute now**.

    The adapter collects information from the appropriate database (App-V Management Server database for App-V release 4.6, and App-V reporting database for App-V release 5.0 and later), packages it, and uploads it to the central FlexNet Manager Suite (or, if it uploads to a parent inventory beacon, the file is briefly saved in `%CommonAppData%\Flexera Software\Incoming\RemoteApplications` on the parent inventory beacon). Allow time for this process to complete before continuing with the next setup procedure.

Import Evidence and Recognize Applications

Because your App-V package naming may be unique to your enterprise, you may need to identify the applications they contain.

If you have many App-V packages, setting up application recognition may be a significant effort on the first import from your App-V server adapter. Once the initial work is done, it is a simpler task to maintain recognition as new App-V packages are put into production.

This procedure continues from the work just completed at the inventory beacon where the adapter runs. Move now to the web interface for FlexNet Manager Suite.

To import evidence for application recognition:

1. Ensure that the upload of data from the App-V server adapter is complete:

   a. In the **Management** view, navigate to the system menu (in the top right corner) and select **Data Inputs**.

      The **Data Inputs** page displays.

   b. Select the **Inventory Data** tab, and select the **Show details** check box near to top.

   c. Find the App-V server adapter listed for the appropriate inventory beacon, and check its **Last import** date, **Duration**, **Validation issues**, and **Status**.

      When these are appropriate (in particular, **Status** is `Successful` on the appropriate **Last import** date), continue this procedure. Until then, you need either to remediate any upload problems, or simply wait until the upload is completed.

      Tip: *This page does not dynamically update results, but shows the status when the page was opened.*

Therefore, if you are waiting for an upload to finish, refresh the page (F5) from time to time to see updated information.

For App-V 5.0 and later, the upload of the `.raa` file saves the contents into staging tables in the compliance database, awaiting the arrival of usage information. For both App-V 4.6 and 5.0 (and later), the upload of the datafile from the App-V server adapter (on an inventory beacon) queues a job with the batch scheduler. When this job can next be processed, the adapter data (which, for App-V 5.0 and later, is combined in this process with the installer evidence from the `.raa` file) is imported into the compliance database. Thereafter, either of two results applies:

- Recognized installer evidence is linked to the application, and shows an installation count for each device on which the evidence was found (and can be further examined on the properties for each of those devices).

  Tip: *The next step, showing consumption against an appropriate license, requires both that the application is linked to the license, and that a reconcile has been run, either automatically on schedule or manually.*

- The installer evidence that was successfully imported but *not* matched to an application record needs your attention to map it to the correct application. (This is more common with the adapter for release 4.6.) To do that, continue with this process.

2. Identify newly-imported evidence (the App-V packages) that requires a link to application records:

   a. Navigate to **License Compliance > Discovered Evidence** (in the **Evidence** group).

      The **Discovered Evidence** page displays. Ensure that the default **Installer evidence** tab is selected.

   b. Near the top of the tab, click **Add filter**, and from the drop-down list select `Type`.

      A second drop-down list appears, listing the possible values of the evidence type.

   c. Select `App-V`.

   d. Click **Add filter** again, and from the pull-down drop-down list select `Assigned`, then choose the value `No`. With both filters defined, click the blue check mark (tick) to apply this filter.

      The list is redrawn to show only evidence of type App-V (the list of App-V packages discovered through the adapter) that have not been matched (manually or automatically) to an application.

      Tip: *You may have no results when these filters are applied. This is a healthy state, meaning that all your App-V evidence is successfully matched to applications. When you have no records here, and want further validation of success, you can drill down into the properties of devices that have a record of installation for the appropriate application.*

3. Use your special knowledge of the App-V packages to link each piece of unassigned evidence to an application record. You may do this either by:

   - Clicking the **Name** of the evidence, which opens the property sheet for this evidence where you can work on the **Applications** tab

   - Following these guidelines to edit locally, still on the **Discovered Evidence** page:

     a. Click anywhere else in a row (other than on the **Name**) to select that App-V package from the list.

        The action buttons above the list become active.

b. Click **Assign**.

A blue editor **Assign evidence to an application** appears above the tabs.

c. Click in the search field, optionally enter a few characters from the application name, and click **Search**.

The editing area expands to include a list of application results matching your search. These applications include any previously created in your enterprise, together with all matching applications from the Application Recognition Library regularly updated by Flexera.

d. Select your chosen application from the list, and (above the list) click **Add**.

The search field is updated to show the name of the application you selected.

Tip: *If you cannot find the correct application in the search results, you may create a new application record by clicking* ***Create an application****, and completing the details in the application properties (the App-V package is automatically listed in the* ***Evidence*** *tab of the application properties).*

e. Click **Assign**.

- The blue editing area closes.
- The App-V package is linked to the application you chose (the App-V package now functions like installer evidence to show consumption against any license linked to the application).
- In the list of evidence, the **Assigned** column is updated to `Yes`, showing that this package has been linked (or assigned) to an application. (If you currently have a filter on the **Assigned** column to show only rows with a `No` value, the evidence you just assigned must disappear from the list.)

Tip: *This links the evidence to the application as an exact match across publisher, name, and version records. To protect against future upgrades of the App-V package, you may wish to generalize the version number with a `%` wild-card. For example, if the original version was `1.0`, manually editing the* ***Version*** *property of the installer evidence to `1.%` means the link to the application remains valid through all the minor upgrades of this package.*

4. If the selected application is not yet linked to a license, you can:

a. Double-click the App-V package in the evidence list (or, while it is selected, click **Open**).

The evidence property sheet opens.

b. Select the **Applications** tab.

c. In the list of applications, click the name of the one you have just assigned to the App-V package (or double-click elsewhere in the row; or select the row and click **Open**).

The application property sheet opens.

d. Select the **Licenses** tab in the application properties.

e. Optionally enter a few characters of a license name (where you know it); click **Search**.

The search area expands to show the list of available licenses (matching any characters you entered).

f. Select the appropriate license from those offered, and click **Add license**.

Where a suitable license does not already exist, you may instead create one (for example, navigate to

License Compliance > Create a license (in the **Licenses** column).

Tip: *Don't forget to link some purchase records to the license to provide your entitlement count.*

3

Issues and Limitations

Diagnosis is covered in this chapter, along with limitations and known issues.

Limitations

The following limitations apply to the current releases of the App-V server adapter:

- In the unlikely event that App-V packages have been shared to non-persistent Citrix Virtual Desktops (formerly XenDesktop) VDI devices (instead of users), and FlexNet Manager Suite is linked to a Citrix Virtual Desktops broker, no license consumption occurs for those non-persistent devices (because non-persistent VDI devices are not modeled within FlexNet Manager Suite when Citrix Virtual Desktops broker information is available).
- License consumption calculations depend on the integration of data both from Active Directory, and from the appropriate App-V database (App-V Management Server database for release 4.6, or reporting database for release 5.0 and later) and the `AppVMgmtSvr.ps1` PowerShell script. As linking of this data occurs when the App-V server adapter data is imported, current users, computers, and groups must be imported from Active Directory *first*, before the latest App-V server adapter import occurs. From FlexNet Manager Suite 2014 R2, this ordering is automatic, since Active Directory data is imported to the central database as soon as it is uploaded from an inventory beacon.

  Tip: *If you have App-V applications secured by security groups from multiple Active Directory domains, ensure that the Active Directory import runs against all applicable domains in your environment. The simplest approach may well be to ensure that you import from all your Active Directory domains, since if you use foreign security principals from multiple trusted domains, it can be difficult to keep track of access to App-V packages. FlexNet Manager Suite imports only from each individually specified Active Directory domain; so you need to ensure that all applicable domains are specified. As an example of multiple domains being affected:*

  - *Suppose you have Group-A in Domain-A that contains a child Group-B, where Group-B actually comes from Domain-B.*
  - *In this case, granting access to an App-V package to Group-A also grants access to Group-B (because of the parent-child relationship between the groups).*
  - *This inheritance continues to work even when there is only one-way trust from Domain-B to Domain-A.*
  - *In such a case, it is imperative that you run an Active Directory import against both Domain-A and Domain-B. When you have many domains, the simplest path is just to run an Active Directory import from every domain.*

- The quality of installer evidence recovered from App-V release 4.6 is not high. You should expect to do remedial work both to generalize the evidence found (for example, using the `%` wild card to generalize version numbering), and to link the evidence to appropriate applications.
- For App-V release 5.0 and later, the system supports installation of the `AppVMgmtSvr.ps1` PowerShell script on only one App-V Management Server. This single Management Server may support multiple Publishing Servers (if necessary spread worldwide for faster distribution of App-V packages to App-V clients); and the App-V clients may report to multiple Reporting Servers (independent of the source from which the App-V packages were downloaded). Different App-V Management Servers do not self-identify in the `.raa` inventory file, and the App-V reporting database does not identify which application usage information is associated with which App-V Management Server. For these reasons, only a single App-V Management Server (for release 5.0 and later) is supported.

Investigating Issues

Proof that the App-V server adapter is operational, and the data upload is also successful, can be seen in either or both of two ways:

- Installation records for the appropriate applications against expected inventory devices (possibly including `Remote` devices for which no hardware inventory is available)
- The presence of any newly-discovered inventory of type `App-V` in the **Discovered Inventory** list, typically with an **Assigned** value of `No`.

If neither of these is the case, the issues could be with:

- Imports from the App-V server adapter on an inventory beacon (for both App-V release 4.6, and App-V release 5.0 or later)
- Imports of the `.raa` file from your App-V Management Server (only for App-V release 5.0 or later)
- Missing links between the installer evidence (representing App-V packages) and the applications in the packages
- Missing consumption on an appropriate license.

Each of these is covered in turn below.

No data imports from adapter on inventory beacon

Check the following to identify the problem with imports from the App-V server adapter:

1. Are you sure that there should be new records? Have new packages been brought into production since the last time inventory was collected and fully processed?
2. Check the **Status** of the latest upload (navigate to the system menu (⚙ ▾ in the top right corner), select **Data Inputs**, choose the **Inventory Data** tab and select **Show details**). Also validate that the **Last import** date is as expected, so that the upload occurred *after* new App-V packages were brought into production.
3. If uploads are not happening, move to the inventory beacon where the adapter runs, and check the status of the App-V connection. Use the **Test connection** button to ensure it can connect to the appropriate database (App-V Management Server database for App-V release 4.6, and the App-V reporting database for release 5.0 and later). Details about setting the connection are in Configuring the Adapter.

4. On the same inventory beacon, test its connection to the central application server for FlexNet Manager Suite, using the **Parent connection** page and the **Test connection** button there. (If this inventory beacon is part of a hierarchy, check the connections all the way up the hierarchy to prove that uploads can reach the central server.)

5. Check for stalled uploads by looking for an App-V inventory file in the `%CommonAppData%\Flexera Software\Beacon\IntermediateData` directory on the inventory beacon (or, in a hierarchy, in the chain of inventory beacons). (Notice that the folder for files from the App-V server adapter is separate from the folder for `.raa` files from the PowerShell scripts used with release 5.0 and later.) App-V data files are named in part for the connection you established (see Configuring the Adapter). Once an inventory file of this type is successfully uploaded, it is removed from this intermediate data location on the inventory beacon; so any file in this folder on any server in the hierarchy has not yet been uploaded to the parent server. Upload failures may occur for temporary reasons, such as a network timeout; but there is a catch-up task run overnight to re-attempt uploads of any stalled files.

6. Has a reconciliation calculation occurred since the upload? Until this occurs (normally overnight), new App-V inventory cannot be displayed in the web interface for FlexNet Manager Suite. Check the date and time on the **System Health** page (⚙ ▼ > **System Health**), in the **License reconciliation** card. The last import and reconciliation must be after the latest upload from the inventory beacon.

7. If you are using App-V release 5.0 or later, a failure to upload and save `.raa` files may also prevent presentation of results, even when the application usage information is successfully imported from the App-V server adapter. Issues with `.raa` files are covered in the next section.

No data imports from the PowerShell script for App-V release 5.0 and later

If uploads of `.raa` files do not appear to be working:

1. Ensure that a new upload is required: that is, that the `AppVMgmtSvr.ps1` script has executed successfully since the last inventory import and license consumption calculation (the most recent file is always saved on the App-V Management Server for checking, and is only replaced the next time that the script executes):
   - Check your scheduling for the script's execution, in particular that the command line options are correct (see Obtaining (and Deploying) the Adapter Components).
   - Check the log file for the last run (you may have renamed or relocated the log file, as described in Command Line for PowerShell Script). In particular, ensure that there were no problems with the file upload to the inventory beacon.

2. Move to that inventory beacon, and check for stalled uploads by looking for an `.raa` file in `%CommonAppData%\Flexera Software\Incoming\RemoteApplications` (this file path is different from the one used for files uploaded by the App-V server adapter). If you have a hierarchy of inventory beacons, check each in turn.

3. If file uploads are happening successfully, has there been a reconcile since the last `.raa` file was uploaded? Check the date and time on the **System Health** page (⚙ ▼ > **System Health**), in the **License reconciliation** card. The last import and reconciliation must be after the latest upload from the inventory beacon. If not, you may (as an administrator) manually trigger a reconcile (navigate to **License Compliance > Reconcile**).

Missing application recognition

For App-V release 4.6, data imported from the App-V server adapter includes only the App-V package name and the Active Directory groups (or individuals) that may access the package, according to the Access Control Lists (ACL). Specifically, this imported information cannot recognize what application is hidden within the App-V package.

Application recognition requires a separate step. Even for App-V release 5.0 and later, where much better installer evidence allows automatic matching of the evidence rules for the appropriate application, there may be cases where the installer evidence needs manual attention. This will be the case when:

- There is no matching application available within FlexNet Manager Suite, either in application records that you have created locally, or in the Application Recognition Library
- There is an appropriate application, but the values returned from App-V do not match with the existing inventory rules for the application record.

In cases where installer evidence from App-V is unmatched, you can check as follows:

1. First be sure that data uploads and imports are happening, as validated in the previous sections.
2. Navigate to **License Compliance** > **Evidence** column > **Discovered Evidence** > **Installer evidence** tab, filter for **Type**=`App-V`, and check the **Assigned** column. If it displays `No`, you need to link this package to an application, as described in Import Evidence and Recognize Applications.) For App-V release 4.6, newly imported evidence is always unassigned, and requires you to manually associate the evidence (or App-V package) with an application.
3. For App-V release 5.0 (and later), when installer evidence appears in the above listing, it may be worth checking why the data from the App-V installer evidence did not match existing evidence rules for the appropriate application (assuming the application is already present locally or in the Application Recognition Library). To do this, navigate to the **Evidence** tab of the application's properties, where all existing evidence "rules" are listed (making sure that **Installer** is the selected subtab). Compare the data displayed there with content in your `.raa` file (see sample `.raa` file in File Format for .raa). Here is a worked example of successful matching using the FileZilla 3 application:

App-V element's attribute	Application's installer evidence property
`msiDisplayName="FileZilla Client 3.2.4.1"`	**Name** `FileZilla Client 3.2.%`
`msiPublisher=""`	**Publisher** (blank)
`msiVersion="3.2.4.1"`	**Version** `3.2.%`
`accessModeID="2"` (produces the evidence type `App-V`)	**Type** Any

Thus the `.raa` entry produces installer evidence that is immediately matched by the existing installer evidence rule for the application, and produces an installation count against that application. But looking ahead (in imagination) to the day when the `.raa` entry covers a version of `4.2.3.1`, the App-V package data in the `.raa` file would no longer match this evidence rule. At that time, if a new rule was yet to be published in the Application Recognition Library, the installer evidence created from the `.raa` file would appear in the **Discovered Evidence** listing, and you could link it to the application, preferably generalizing it (similarly to the example above) to create a rule that would match several minor releases.

Missing consumption

Showing consumption of license entitlements for App-V packages (and the applications they contain) requires:

- The adapter gathers data from the App-V server and uploads and imports it into FlexNet Manager Suite (see first section above for more details).
- For App-V release 5.0 or later, the `.raa` file is uploaded from your App-V Management Server

- The application is recognized, either automatically, or because you have linked the App-V package to an application record (see previous section)
- The application is linked to a license (and in turn the license should be linked to purchase records to show your legal entitlements) — here, this is left as an exercise for the reader.
- Active Directory imports are current, allowing mapping of the groups and users from the ACLs in the App-V Management Server database to user records in FlexNet Manager Suite.

These notes address the last stage, enabling Active Directory to map from the ACL lists to user records. This process is automatic, provided that all the necessary data is available.

1. On the appropriate inventory beacon, open the **Active Directory** page (from the **Connections** group), and validate that a connection is both established and scheduled. (For details, see *Importing from Active Directory* in the online help.) Review the **Last run** time to see when data was last collected, uploaded, and imported. You may also choose **Execute Now** if imports have been disrupted.
2. Once sufficient time has passed for Active Directory data collection, upload, and import (normally, 30 minutes should be more than adequate), review the list of **All Users** to establish that the expected user names are all available (navigate to **Enterprise > All Users**).

Tip: *If you have App-V applications secured by security groups from multiple Active Directory domains, ensure that the Active Directory import runs against all applicable domains in your environment. The simplest approach may well be to ensure that you import from all your Active Directory domains, since if you use foreign security principals from multiple trusted domains, it can be difficult to keep track of access to App-V packages. FlexNet Manager Suite imports only from each individually specified Active Directory domain; so you need to ensure that all applicable domains are specified. As an example of multiple domains being affected:*

- *Suppose you have Group-A in Domain-A that contains a child Group-B, where Group-B actually comes from Domain-B.*
- *In this case, granting access to an App-V package to Group-A also grants access to Group-B (because of the parent-child relationship between the groups).*
- *This inheritance continues to work even when there is only one-way trust from Domain-B to Domain-A.*
- *In such a case, it is imperative that you run an Active Directory import against both Domain-A and Domain-B. When you have many domains, the simplest path is just to run an Active Directory import from every domain.*

Note: *You cannot review Active Directory group memberships within FlexNet Manager Suite. Only the resulting list of users is available (along with computers, sites, and subnets).*

Known Issues

The following issues relate primarily to the stability of data available for collection from the App-V server.

Renaming packages

App-V allows you to rename packages (without necessarily changing their contents). When a package is renamed, the App-V application record no longer links to usage records within the appropriate App-V database. This means that the App-V server adapter (and, for App-V release 5.0 and later, the `AppVMgmtSvr.ps1` PowerShell script) cannot import

meaningful data for license consumption calculations. As well, the installer evidence generated from the App-V import now has a different name, which may not match evidence rules for the appropriate application. If that is the case, the consumption calculations for any license linked to an application that was linked to the previous installer evidence from App-V drops to zero (from this import connection).

For this reason, it is *strongly recommended* that you do not change the name of existing App-V packages in the App-V Management Server interface. Where renaming is unavoidable, remember that you may need to link the new package (represented as installer evidence in FlexNet Manager Suite) to the application to restore consumption calculations (especially for App-V release 4.6).

Updating package versions

Typical "rules" for applying installer evidence to identify applications use the publisher, the application, and the version recorded in the installer evidence. If you use a version number for your App-V packages (such as 1.0), and link exactly this App-V 'installer' evidence to the application record in FlexNet Manager Suite, then updating the version number of the package on your App-V Server (say, to 1.1) may break the link to the application, and the linked license loses its consumption as a result.

You can avoid this problem (at least for minor version changes) by using wild-cards in the linking of evidence to applications. To do this, after you have linked the App-V evidence to the application:

1. Navigate to the **Installer Evidence Properties** page.
2. Select the **General** tab.
3. Edit the **Version** property, using a wild-card percentage sign (`%`) to generalize the match across multiple versions. For example:
   - The original (say `9.2`) is an exact match for only one version number.
   - A value of `9.%` matches all the minor releases of version 9.
   - A value of `%` matches all versions of the App-V package with the same name and publisher.

4

Data mapping

This chapter covers the relationships between the data fields in the source App-V data and the final locations of the data inside the FlexNet Manager Suite database.

App-V Release 4.6 Data Transfers

This table lists the data extracted from App-V release 4.6, and where it is stored the compliance database in FlexNet Manager Suite. Imported data is stored in temporary staging tables for data manipulation, and then moved into their final destination tables as noted below. These columns in the compliance database are available for use in custom reports, and the like.

App-V (Table)/Column	FNMS (Table)/Column	Notes
(REPORTING_CLIENT_INFORMATION) host_name	(ComplianceComputer) ComputerName	If the computer name already exists in the ComplianceComputer table, this device is identified as the consuming device. If it does not already exist, the device is added to the ComplianceComputer table with a ComplianceComputerTypeID of 4, which (through the ComplianceComputerType table) indicates a remote device from which inventory cannot be collected. All such created records display their connection name (as the Inventory Agent), are marked as incomplete records, and are given a fictitious serial number of 1.
(REPORTING_CLIENT_INFORMATION) host_name	(ComplianceComputer) ComplianceDomainID	The domain name is extracted from the host record on the App-V server. If the domain does not already exist in the compliance database, it is added to the ComplianceDomain table, and as required the ComplianceDomainID may be updated in the ComplianceComputer table.

App-V (Table)/Column	FNMS (Table)/Column	Notes
(APPLICATION_USAGE) username	(ComplianceComputer) ComplianceUserID	The last logged on user for this computer.
(VW_APPLICATIONS) app_name	(InstallerEvidence) DisplayName	The name of the App-V package (which may bear no resemblance to the application inside) is used to identify the evidence record. ***Note:*** *It is possible for an App-V package to contain more than one application. This makes it impossible to link the App-V evidence to a single application record. Best practice is to make each package contain exactly one licensable application.*
(VW_APPLICATIONS) version	(InstallerEvidence) Version	The version of the App-V package. Again, this bears no necessary relationship to the version of the application inside the package, and is at the whim of the person preparing the package (for example, it may represent the number of times the application package was modified).
String literal App-V	(InstallerEvidence) Publisher	All App-V evidence is given the publisher name of App-V.

App-V Release 5.0 (and Later) Data Transfers

This data is imported from Microsoft App-V release 5.0 and later. It is imported by the combination of the PowerShell script installed on your App-v Management Server, and the App-V server adapter on an inventory beacon that can access the App-V reporting database.

Several staging tables in the compliance database store information uploaded from the App-V reporting database and the .raa files. These include:

- ImportedComputer
- ImportedInstalledInstallerEvidenceUsage
- ImportedInstallerEvidence
- ImportedRemoteApplication
- ImportedRemoteApplicationAccess
- ImportedRemoteApplicationServer
- ImportedRemoteApplicationInstallerData.

Data is held in these tables until the import from the App-V server adapter is complete, and then the resolvers are triggered to combine the data from:

- The most recent Active Directory import(s) across all relevant domains
- The most recent imports of the `.raa` file from all App-V Management Server
- The App-V reporting database.

For this reason, it is important that the import from the App-V reporting database happens last.

The following table lists the elements and their attributes from the `.raa` file, and their final table and column in the compliance database within FlexNet Manager Suite.

XML (Element) / Attribute	FNMS (Table) / Column	Notes
`(app) appID`	n.a.	Used as a key as required across the temporary tables listed above.
`(app) userSid`	n.a.	The group SID (from Active Directory) identifying the users and computers with access to the package. This drives the linking of inventory device and user records to the application.
`(msiData) msiDisplayName`	`(InstallerEvidence) DisplayName`	Visible as the **Name** in Installer Evidence properties.
`(msiData) msiPublisher`	`(InstallerEvidence) Publisher`	Visible as the **Publisher** in Installer Evidence properties.
`(msiData) msiVersion`	`(InstallerEvidence) Version`	Visible as the **Version** in Installer Evidence properties.
`(msiData) msiProductCode`	`(InstallerEvidence) ProductCode`	Not visible in the web interface.

The following are the views used for source data from the App-V reporting database. It is not possible to give a simple mapping of this source data to columns in particular database tables within FlexNet Manager Suite, because the data is normalized and otherwise processed at each stage. For example, the `username`column collected from `view_ApplicationUsage` in the App-V reporting database is already subject to considerable validation and processing before it is stored in the `lastloggedonuser` column in the staging tables (staging tables are listed above). Thereafter, the user name is correlated with other inventory sources, resulting in further processing before it is used to determine a link between the application and the user. For that reason, this table shows only the source content in the App-V reporting database.

App-V View	Columns
dbo.view_ApplicationUsage	• app_name • app_version • end_time • host_id • start_time • username • version_guid
dbo.view_ClientInformation	• host_id • host_name
dbo.view_PackageInformation	• host_id • package_id • version_guid

III

BMC Atrium and Remedy Integration

The BMC Atrium adapter is a mechanism for two-way information synchronization between an on-premises implementation of FlexNet Manager Suite, and BMC Software's Atrium configuration management database (CMDB) and Remedy IT Services Management product. Data flows in two directions, which in this document are described from the viewpoint of FlexNet Manager Suite as export and import:

- The adapter exports computer system and recognized product information from FlexNet Manager Suite to the CMDB. (Details of fields exported are available in Appendix 1: Export of Computers and Appendix 2: Export of Applications/Products.)
- The adapter also imports Asset role, status and owner information from the CMDB and Remedy into FlexNet Manager Suite. (Details of the imported data are available in Appendix 3: Import of Assets.)

Note: *This adapter is only available for on-premises implementations of FlexNet Manager Suite. Specifically, it cannot be used with the cloud implementation.*

This documentation (and the BMC Atrium adapter) assume that the Atrium server is running on a Microsoft Windows operating system.

This documentation covers the following supported products:

- FlexNet Manager Suite Version 2019 R2
- BMC Atrium CMDB version 8.1 through 19.02 (together with Atrium Integrator version 8.1 through 9.1)
- BMC Remedy ITSM Version 7.6.04 SP4–9.1

1

Architecture, Operation, and Prerequisites

This chapter provides a framework for your understanding of later, more detailed information.

Architecture and Operations

To facilitate the two-way exchange of data between FlexNet Manager Suite and BMC Atrium, components already supplied with both systems are used, and additional elements are supplied with the downloaded adapter.

The following diagram combines the positioning of these elements on different servers, with the data flows for both export and import (as defined in BMC Atrium and Remedy Integration). There are two determining factors in this structural arrangement:

- The schedule for triggering both exports and imports is configured within Atrium, and the Atrium Integrator needs to invoke the data export utility and the FlexNet Business Importer. For this reason, all three components (Atrium Integrator, export utility, and Business Importer) are located on the same server.
- For performance, it is preferable that the intermediate database be on the same database server as your compliance database for FlexNet Manager Suite. (Where you choose a different location, ensure there is a high-speed network connection between the two servers, and a trust relationship for the accounts used during operations.)

The combination of these two determinants produces the following physical architecture. Net data flows are shown here, with each process also shown separately below.

Once the adapter components are installed and all systems are configured (as described in the chapter Installing the Adapter), operation is fully automatic, triggered by schedules you set during configuration. The two directions of data exchange function independently, and are summarized below.

Export of Computers and Products

In summary, the process for exporting computer and product information from FlexNet Manager Suite to BMC Atrium (shown below in blue arrows) is as follows:

1. The schedule for export fires in Atrium, and the Atrium Integrator calls the data export tool (installed as part of the adapter).
2. The data export tool reads content from the compliance database for FlexNet Manager Suite, and writes the data into the intermediate database. (It also creates entries in the log file, and sends alert emails if there any errors.)
3. On success, the Atrium Integrator reads the data from the intermediate database, maps it using the defined data transforms to the fields required in Atrium

4. The Atrium Integrator writes the results into the CMDB.

Import of Assets

In summary, the process for importing assets (and their ownership) from BMC Atrium into FlexNet Manager Suite (shown below in green arrows) is as follows:

1. The schedule for import fires in Atrium.
2. The Atrium Integrator reads data from both the Remedy database and the CMDB, and writes the results into the Intermediate database. (It also sends alert emails if there are any errors.)
3. On success, the Atrium Integrator calls the FlexNet Business Importer.
4. The Business Importer reads the data from the intermediate database, transforms it as required, and writes the results into the compliance database of FlexNet Manager Suite.

Prerequisites

The prerequisites for the BMC Atrium adapter include the following:

- On the server where the data utility is to be installed, .NET Framework 4.5 is required. (To validate, open Windows **Programs and Features**, and search for `.NET`.)
- FlexNet Business Importer must be installed on the same server where the Atrium Integrator is installed. This is installed as a part of the FlexNet Business Adapter Studio. Collecting the installer files is covered in Obtaining the Adapter Components, and completing the installation and configuration is detailed in Install the Business Importer and Link with Remedy.
- The adapter requires an intermediate database to be installed, conceptually between the compliance database for FlexNet Manager Suite and the CMDB for BMC Remedy. In practice, this intermediate database may reside on the same database server as the compliance database; or any other convenient server running at least Microsoft SQL Server 2012 that is accessible from both your BMC Atrium server and your FlexNet Manager Suite database server. Scripts to set up the database are included with the adapter, and the process is covered in Preparing the Databases.

- FlexNet Manager Suite Version 2019 R2
- A FlexNet Manager Platform BMC Atrium CMDB Option license. To verify that you have this license term, navigate to the system menu (⚙ ▼ in the top right corner) > **FlexNet Manager Suite License**, and look for this item in the list of **Licensed products**. If not present, you can request the license term through the `Contact us to renew or upgrade` email link on the same page.
- BMC Atrium CMDB version 8.1 through 19.02 (together with Atrium Integrator version 8.1 through 9.1)
- BMC Remedy ITSM Version 7.6.04 SP4–9.1

Obtaining the Adapter Components

It is most convenient to collect the adapter, and the main elements it contains is to complete this procedure on the computer where you will install the adapter (see Architecture and Operations).

To download adapter components:

1. Use your browser to access the Flexera Customer Community.
   a. On https://flexeracommunity.force.com/customer/CCLanding, use the account details emailed to you with your order confirmation from Flexera to log in (using the **Login** link in the top right).

      Tip: *Access requires your Customer Community user name and password. If you do not have one, use the `Request Community Access` link on the login page to request one. Your credentials are configured for access to content you have licensed.*

   b. Select the **Downloads** tab from the row across the top of the page.

      A routing page appears to let you `Access Product and License Center`, displaying lists of products from Flexera.

   c. In the lists of products, identify `FlexNet Manager Platform`, and immediately below it, click **Access Above Products**.

      The `Product and License Center` site is displayed.

   d. In the `Your Downloads` section of the `Home` page, click the link for FlexNet Manager Platform.

   e. In the `Download Packages` page, click the link for FlexNet Manager Platform 2019 R2 to access the downloads. (You may need to repeat this action on a second page to access the downloadable files.)
2. Select the `Adapter Tools for` FlexNet Manager Suite `2019 R2` archive (`Adapter Tools for` FlexNet Manager Suite `2019 R2.zip`), and save to a convenient location (such as `C:\temp`) on the server where Atrium Integrator Spoon is installed.
3. Similarly, download `Business Adapter Studio for` FlexNet Manager Suite *`version.zip`* and save on the same server.

   Installation of the Business Adapter Studio is covered in Install the Business Importer and Link with Remedy.
4. Unzip the adapter tools archive, navigate through `Tier 1 Adapter Tools` > `BMCAtriumAdapter`, and copy the

contents into `C:\Program Files\Flexera\`.

Important: *This path is referenced in various steps in the Atrium jobs and transformations (see file list below). If you do not use the same path, you must edit the Atrium Integrator Spoon jobs and transformations manually to point the shell scripts to different location.*

The archive includes the following key components:

- An SQL script to create the intermediate staging database (*archive*`\BMCAtriumAdapter\scripts\CreateIntermediateDatabase.sql`)
- Another SQL script to modify the database for FlexNet Manager Suite (*archive*`\BMCAtriumAdapter\scripts\FNMPDatabaseChanges.sql`)
- The definition of Atrium Integrator jobs (`.kjb`) and transformations (`.ktr`):
  - *archive*`\BMCAtriumAdapter\BMCAtriumAdapter.kjb`
  - *archive*`\BMCAtriumAdapter\FNMPExports\Execute FNMP Export Tool.kjb`
  - *archive*`\BMCAtriumAdapter\FNMPExports\ExportComputerProductRelationshipsTransform.ktr`
  - *archive*`\BMCAtriumAdapter\FNMPExports\ExportComputersTransform.ktr`
  - *archive*`\BMCAtriumAdapter\FNMPExports\ExportProductsTransform.ktr`
  - *archive*`\BMCAtriumAdapter\FNMPExports\ExportUpdateIntermediateDBTransform.ktr`
  - *archive*`\BMCAtriumAdapter\FNMPExports\FNMPExports.kjb`
  - *archive*`\BMCAtriumAdapter\FNMPImports\Asset.ktr`
  - *archive*`\BMCAtriumAdapter\FNMPImports\ImportUpdateIntermediateDBTransform.ktr`
- The adapter's main executable (*archive*`\BMCAtriumAdapter\FNMPExports\Utility\FNMPDataExportUtility.exe`), and its many dependencies, including:
  - Query files (*archive*`\BMCAtriumAdapter\FNMPExports\Utility\Queries\Computers.txt`, `ProductIds.txt`, `Products.txt`)
  - Adapter configuration file (*archive*`\BMCAtriumAdapter\FNMPExports\Utility\FNMPDataExportUtility.exe.config`)
- Business Importer adapter definition file (*archive*`\BMCAtriumAdapter\FNMPImports\FNMPImportDefinition\Asset.xml`).

2

Installing the Adapter

Integrating two sophisticated systems likes these involves a number of procedures, documented in the topics shown:

1. Installing the new, intermediate DB used for data manipulation and transfer (see Preparing the Databases).
2. Modify the compliance database in FlexNet Manager Suite with a custom property for additional values (also in Preparing the Databases).
3. Installing and configuring the export utility that brings data out of FlexNet Manager Suite and into the intermediate database (still in Preparing the Databases).
4. Declaring a dataset within Atrium for data exchange, and authorizing an account to use that dataset (see Reserving an Atrium Dataset).
5. Importing the supplied Atrium jobs and data transforms into your Atrium system (see Importing Atrium Jobs and Transforms).
6. Configuring the supplied Atrium jobs with appropriate settings for your environment (see Configure Atrium Jobs).
7. For the import of asset data from Atrium to FlexNet Manager Suite, installing the FlexNet Business Adapter Studio (which brings with it the Business Importer), and integrating the Business Importer with BMC Remedy Mid-Tier (see Install the Business Importer and Link with Remedy).
8. Initially populating the CMBD with configuration items from FlexNet Manager Suite, and then adjusting settings to allow for reuse for future imports (see Tuning Creation of Configuration Items).
9. Schedule regular operations of the adapter (see Scheduling the Adapter).

This chapter also includes details about Verifying Data Export to BMC Atrium.

If you are transferring large datasets, you should also prepare an environment variable as described in Preparing for Large Datasets. (Other configuration options, including relocating or renaming the log file, are also documented in the chapter Additional Customization of the Adapter.)

Preparing the Databases

You have a clear understanding of your physical architecture for the installation (see Architecture and Operations), and have downloaded and unzipped the archive containing the adapter (see Obtaining the Adapter Components).

To prepare the databases:

1. Determine where the intermediate database used by the adapter will reside, and log into that database server using an account that has DB Owner privileges.

   For details, refer back to Architecture and Operations.

   Tip: *If the archive cannot be accessed from this database server, first copy the script (or both scripts for a common DB server) from this folder to a location accessible from SQL Server Studio. If you use copies of the scripts, modify the paths described below to suit.*

2. On this server, in Microsoft SQL Server Studio, create the intermediate database (suggested name `AtriumIntermediateDB`).

3. Open a query window that is connected to your new database, and execute `CreateIntermediateDatabase.sql` (located in the unzipped archive, by default in `C:\Program Files\Flexera\BMCAtriumAdapter\scripts`).

   This SQL procedure creates the required tables and pre-populates some fixed values.

4. If the intermediate database is on another server separate from your compliance database for FlexNet Manager Suite, switch over and log into the FlexNet Manager Suite database server, again using an account that has DB Owner privileges.

   You need access to the second SQL script here, so if necessary, copy the second script to this server before continuing.

5. In Microsoft SQL Server Studio, navigate to and execute `C:\Program Files\Flexera\BMCAtriumAdapter\scripts\FNMPDatabaseChanges.sql`.

   This SQL script adds a custom property to the **Details** tab of computer properties in FlexNet Manager Suite where the machine's **Role** can be imported from BMC Atrium.

6. On the server where the data extraction executable will run (likely, the Atrium server), open a command prompt, and navigate to `C:\ProgramFiles\Flexera\BMCAtriumAdapter\FNMPExports\Utility`.

   This executable requires .NET Framework 4.5 installed, a high-speed network access to the database(s), and an account trusted to access both the compliance database and the intermediate database. It must be callable by the Atrium Integrator. If need be, first copy the *entire contents* of the folder shown above (including subdirectories) to a convenient location on the appropriate machine.

   Tip: *To validate that the host in correctly configured with .NET Framework 4.5 or later, execute*

   ```
   FNMPDataExportUtility.exe /help
   ```

   A list of command-line options is displayed. If not, investigate the version of .NET installed.

7. Execute the following (on a single line):

   ```
   FNMPDataExportUtility.exe
                            /Source "FNMP-DBConnectionString"
                            /Target "IntermediateDBConnectionString"
                            /Encrypt true
   ```

where:

Parameter	Notes
FNMP-DBConnectionString	The connection string (enclosed in double quotation marks) for the export utility to connection with the compliance database for FlexNet Manager Suite. The `Data Source` value must be the fully-qualified domain name of the database server. Where the compliance database is not in the default instance on this server, add the instance name (separated by a slash). An example connection string when the default name for the compliance database is FNMSCompliance, is (all on one line): `Data Source=DBServer.example.com/myInstance; Initial Catalog=FNMSCompliance;Integrated Security=SSPI; Connection Timeout=60;Min Pool Size=2;Max Pool Size=20;`
IntermediateDB ConnectionString	The connection string (enclosed in double quotation marks) for the export utility to connection with the intermediate database you recently created. An example connection string when this is on the same server and instance as the compliance database, and the default name for the intermediate database is `IntermediateDB`, is (all on one line): `Data Source=DBServer.example.com/myInstance; Initial Catalog=IntermediateDB;Integrated Security=SSPI; Connection Timeout=60;Min Pool Size=2;Max Pool Size=20;`
`/Encrypt`	Required field. When set to `true` the connection strings are encrypted when they are saved into the configuration file for the export utility. Use `false` when not encrypting the connection strings. (The configuration file lives in the same folder as the executable, and is called `FNMPDataExportUtility.exe.config`).

8. Configure where the export utility sends email notifications about any export errors.

   a. From the same folder, open `FNMPDataExportUtility.exe.config` in a flat text (or XML) editor.

   b. Locate the following section and replace the values shown:

```
<appender name="SmtpAppender" type="log4net.Appender.SmtpAppender">
        <to value="toaddress@somedomain.com"></to>
        <from value="fromaddress@somedomain.com"></from>
        <subject value=" Atrium Integration Adapter Error"></subject>
        <smtpHost value="smtp.somedomain.com"></smtpHost>
```

   where:

Value	Notes
toaddress @somedomain.com	Valid email address for the person who is to receive alerts when errors occur during export.

Value	Notes
fromaddress @somedomain.com	Valid email address for an account known on the specified SMTP host that is identified as the sender of the email when an error occurs.
smtp.somedomain.com	The fully qualified domain name of the SMTP email server.

c. While here, you may want to customize the location of the log file for the export utility:

```
  <log4net>
    <appender name="RollingFileAppender"
type="log4net.Appender.RollingFileAppender">
      <file value="C:\Log.txt" />
```

d. Save the edited file, and exit the editor.

Creating a View Only SQL User in FlexNet Manager Suite

This task outlines how to create and configure accounts for the SQL user and to promote this user to a FlexNet Manager Suite View only operator. These accounts are required to fetch data from FlexNet Manager Suite using BMC Atrium Integrator Job and to populate the BMC Atrium CMDB Intermediate database.

To prepare the view-only SQL user for FlexNet Manager Suite compliance database, begin by creating a user (for example, fnmsintegrator) for the FNMSCompliance database that can log in using SQL Authentication as follows:

1. Open the Microsoft SQL Server Management Studio.
2. Expand the **Security** folder (first-level child of the top SQL Server node) and right-click **Logins**.
3. Right-click `fnmsintegrator` user and select **Properties** from the context menu.
4. Configure the Login properties for fnmsintegrator user as follows:
   a. Set the Server Role as `public` to grant server-wide security privileges to the user.
   b. In the User Mapping section, select `FNMSCompliance database` for this user.
   c. Select the `public` and `db_datareader` role membership for this user against FNMSCompliance database.
   d. Select `db_datawriter membership` *only* if the fnmsintegrator user is used for populating data to FNMSCompliance database from IntermediateDB.
5. Add the above user to the `ComplianceOperator` group for the FNMSCompliance database. For example, the following SQL query can be used for adding a fnmsintegrator user:

```
Insert into ComplianceOperator(OperatorLogin,IsEnabled,GlobalOperator,Interactive)
values('fnmsintegrator',1,0,0)
```

6. In FlexNet Manager Suite, navigate to the system menu (⚙ ▼ in the top right corner) and choose **Accounts**.
7. Grant `View Only` permissions to the operator that you created in the above steps.
8. Connect to SQL Management Studio as the above user to validate that these read permissions have propagated and run the following query.

```
select * from ComplianceComputerCurrentUser
```

9. If the query does not return results, execute the `SyncPreCalculatedAccessRightsForOperator` stored procedure for this `ComplianceOperator` as follows:
   a. Open Microsoft SQL Server Management Studio.
   b. Login to FNMSCompliance database as Administrator.
   c. Execute the following against corresponding Tenant UID.

```
SetTenantUID '[Tenant UID]'
                            Declare @OutputBit BIT = 1
                            Declare @OperatorID INT = (select
ComplianceOperatorID from ComplianceOperator where OperatorLogin =
'fnmsintegrator ')
                            Exec SyncPreCalculatedAccessRightsForOperator
@OperatorID, 0, @OutputBit
```

For example, the last line may be:

```
EXEC SyncPreCalculatedAccessRightsForOperator 2,0,1
```

10. Retry the `SELECT on ComplianceComputerCurrentUser` to ensure it works correctly.
11. Ensure that the exporter utility config file is using this SQL user in the connection string for the FNMSCompliance database.

Reserving an Atrium Dataset

You must specify a dataset for information exchange, and ensure you have an account with write access to that dataset. This procedure sets up a data area within Atrium reserved for data exchange with the FlexNet Manager Suite adapter. Your Atrium Administrator may assist with an account that had permissions to access this dataset. You will reference this dataset later in the installation of the adapter.

To reserve the area in Atrium for data exchange:

1. In the BMC Remedy User application, open the CMDB console.
2. Select **Reconciliation Manager > Create > Dataset**.

   A **Dataset Information** form appears.

3. Complete the following parameters:

Dataset name	A friendly name for the dataset that you will recognize later in other listings.
Dataset ID	The system identifier for the dataset. ***Tip:*** *This value is needed when you configure your Atrium jobs.*
Accessibility	Use the drop-down list to set this to `Writable`.
DatasetType	Use the drop-down list to set this to `Regular`.

4. Click **Save**.

   The new dataset is created.

5. Consult with your Atrium Administrator about an account with write access to the created dataset. This account must be used on the Atrium side to run the adapter.

For more information about creating the dataset, see http://discovery.bmc.com/confluence/display/81/Configuring+the+BMC+Atrium+Adapter.

Importing Atrium Jobs and Transforms

You need to get the supplied files into the Atrium repository, configured for your local connections. For this procedure, you must be logged into your Atrium server with administrator privileges. This server needs access to the unzipped archive for the adapter (in this procedure, shown as on the same server).

To import jobs and transforms to Atrium:

1. Open the Atrium Integrator Spoon tool.
2. Select **File > Open URL...**, navigate to and open `C:\Program Files\Flexera\BMCAtriumAdapter\FNMPExports\ExportComputersTransform.ktr`.
3. Modify the connection strings for each of:
   - Atrium_CMDB
   - AR Server
   - IntermediateDB.
4. Click **File > Save**.

Repeat this procedure for each of the following files *in the order listed:*

1. C:\Program Files\Flexera\BMCAtriumAdapter\FNMPExports\ExportProductsTransform.ktr
2. C:\Program Files\Flexera\BMCAtriumAdapter\FNMPExports\ExportComputerProductRelationshipsTransform.ktr
3. C:\Program Files\Flexera\BMCAtriumAdapter\FNMPExports\ExportUpdateIntermediateDBTransform.ktr

4. C:\Program Files\Flexera\BMCAtriumAdapter\FNMPImports\ Asset.ktr
5. C:\Program Files\Flexera\BMCAtriumAdapter\FNMPImports\ ImportUpdateIntermediateDBTransform.ktr
6. C:\Program Files\Flexera\BMCAtriumAdapter\FNMPExports\FNMPExports.kjb
7. C:\Program Files\Flexera\BMCAtriumAdapter\FNMPExports\ Execute FNMP Export Tool.kjb
8. C:\Program Files\Flexera\BMCAtriumAdapter\BMCAtriumAdapter.kjb

Configure Atrium Jobs

Once you have completed the import of both tasks and jobs into the Atrium repository (as in Importing Atrium Jobs and Transforms), you must configure the jobs, while still logged into Atrium Spoon, by linking them to transforms.

To configure the jobs in Atrium:

1. Select **File > Open...**, and select `FNMPExports.kjb` from the repository.

   A map of the transformations required for the job is displayed.

2. Right-click `ExportComputersTransform`, and from the context menu select **Edit job entry**.

   The **Job entry details for this transformation** dialog appears. The transform name as recorded in the job entry appears in the first field, **Name of job entry**.

3. Select the **Specify by name and directory** option.

4. To provide a value in the associated field, browse the repository and select the transform name matching the first field.

   For example, when processing the job entry for `ExportComputersTransform`, select the transform of the same name from the repository.

5. Click **OK**.

6. Repeat steps 2-5 for each of the remaining job entries in this chart:

   - `ExportProductsTransform.ktr`
   - `ExportComputerProductRelationshipsTransform.ktr`
   - `ExportUpdateIntermediateDBTransform.ktr`.

7. Select **File > Save**.

8. Select **File > Open...**, and select `BMCAtriumAdapter.kjb` from the repository.

   A map of the job displays.

9. Repeat steps 2-5 for the transform in this chart:

   - Asset.ktr.

10. Double-click the `FNMPExports` job.

    The **Executing a job** dialog appears. The **Job entry name** appears as the first field.

11. Select the **Respository: specify by name** option.
12. To provide a value in the associated field, browse the repository and select the job name matching the first field.
13. Click **OK**.
14. Repeat steps 10-13 for the remaining job in this chart:
    - `Execute FNMP Export Tool.kjb`.
15. Select **File > Save**.
16. Either press `Ctrl-J`; or right-click the `BMCAtriumAdapter` diagram background, and from the context menu select **Job settings**.

    The **Job properties** dialog opens.

17. Edit the following values:

COMPANYNAME	Enter your company name as it is recorded in the CMDB. (A match is required.)
DATASETID	Enter the ID for the dataset that you created for the adapter (see Reserving an Atrium Dataset).

18. Select **File > Save**.

19. To configure email alerts for errors occurring on the Atrium side of the adapter:

    a. In the `BMCAtriumAdapter` job, right-click on the `Mail` step in the center.

    b. From the option menu, select **Edit job entry**.

    The **Job mail details** dialog appears.

    c. In the **Addresses** tab, insert the following:

    - **Destination address** — the person who is to receive the email alert for each error.
    - **Sender name** — the value to appear on the emails as the sender.
    - **Sender address** — The email address from which the email will appear to come. This must be a valid email account known to the mail server (specified next).

    d. Switch to the **Server** tab, and complete the **SMTP Server** details.

    e. Click **OK**.

20. Select **File > Save**.

Install the Business Importer and Link with Remedy

The Business Importer performs parts of the integration work, writing data from Remedy into the FlexNet Manager Suite database. The Business Importer is installed as part of the Business Adapter Studio, a stand-alone environment you can use to build business adapters. These 'adapt' the format of data from other business systems in your computer estate for import into FlexNet Manager Suite, and are exercised by the Business Importer. You downloaded the installer for the Business Adapter Studio in Obtaining the Adapter Components. To enable the business import process, you also configure the reconciliation process to link with the Business Importer. Since the Atrium Integrator must trigger the Business Importer, the normal architecture is to install the Business Adapter Studio on the Atrium server. In this

location, the business adapter for importing asset records must also be amended to update its access to the compliance database.

To install the Business Importer and link with Remedy:

1. On the Atrium server, navigate to the downloaded archive, and extract the `Business Adapter Studio` folder.
2. Execute `Business Adapter Studio\setup.exe`.

   The installation wizard for Business Adapter Studio opens.
3. Complete the installation wizard using the default values it provides.
4. Use the Business Adapter Studio to update connection details for the asset business adapter for both output and input:
   a. On the Atrium server, launch the newly-installed Business Adapter Studio.
   b. Do either of the following:
      - Click the Open icon ( ) in the tool bar.
      - From the **File** menu, click **Open...**.
   c. Navigate to the asset business adapter, by default located at `C:\ProgramFiles\Flexera\BMCAtriumAdapter\FNMPImports\FNMPImportDefinition\Asset.xml`, and click **Open**.
   d. In the navigation bar on the left, click the top-most node, labeled `FlexNet Manager Platform Database`, to display the output connection properties for this adapter.

      Tip: *This node is visible only when the `Business Adapter Studio` is running on a server other than an inventory beacon. (On an inventory beacon, the `Business Adapter Studio` takes the connection details from the inventory beacon, which already has them recorded.)*
   e. In the **Database Connection** panel, select **Use the specific connection information below**, and click the ellipsis (**...**) at the right-hand end of the **Connection String** field to expose the **Data Link Properties** dialog.
   f. Edit the connection details so that they apply to your compliance database:
      a. For step 1, enter the SQL Server instance name for your compliance database. Examples:
         - IPv4 address: `198.51.100.123`
         - The database server hostname: `MyFNMSServer`
         - A non-default database instance: `198.51.100.123\MSInst3`
         - Non-standard connection port: `198.51.100.123,5678`
      b. Supply the credentials for writing to the compliance database. If you are using an account specific to your database server, choose **Use a specific user name and password**, and complete the account name and password in the next two fields.
      c. For **Select the database on the server**, use the drop-down to choose your compliance database (the default name suggested during installation was FNMSCompliance).

   d. Click **Test Connection**, and rejoice at success (or remedy any issues until the connection succeeds).

g. Click **OK** to close the dialog.

h. In the left-hand navigation bar, click on the next row that names the adapter (`Asset`).

i. In the second panel, **Connection and Query Properties**, click the ellipsis (**...**) on the right end of the **Connection** field to expose the relevant **Data Link Properties** dialog for the staging database.

j. In the **Provider** tab, select the database type (typically `Microsoft OLE DB Provider for SQL Server`), and click **Next**.

k. In the same way as before, complete the details in the **Connection** tab, but this time for the staging database, and validate your data with the **Test Connection** button.

l. Click **OK** to close the dialog, and save the updated adapter (`Ctrl+S`, or click the diskette icon in the toolbar, or **File > Save**).

5. Log in to BMC Remedy Mid-Tier using an account name and password provided by your BMC Administrator.

   The **IT Home** page appears.

6. Click the **Applications** tab on the left boundary.

   The Applications menu expands.

7. Select **Atrium Core**, and in the sub-menu select **Atrium Core Console**.

   The **BMC Atrium Core Console** window opens.

8. Click the **Application Launcher** tab on the left edge, then select **Applications** from the menu, and in the sub-menu select **Reconciliation**.

   The **Reconciliation** tab opens.

9. Click the new job icon in the toolbar.

The **Job Editor** page appears.

10. In the **Name** field, provide a name for this job that you will recognize later.

Suggested value: `FNMP Asset Reconciliation`.

11. In the **Activities** group, click **New**.

The **Activities** group changes to the **Edit Activity** area.

12. Complete the details for the first (Identify) activity:

a. In the **Name** field, provide a name you will recognize later.

Suggested value: `Identify`.

b. Ensure that the **Continue on Error** check box is clear.

Recall that the system is configured to send emails on any errors in exporting the assets to FlexNet Manager Suite.

c. Set the **Sequence** number to control ordering of activities in the job.

Recommendation: Number in hundreds to allow for later changes. Suggested value: `500`.

d. In the **Dataset Configuration** area, click the first icon to add a new dataset, and select the one you created for this exchange.

The suggested value was `BMC.FNMS`.

e. Click **Done** to save details of this activity.

The **Activities** group reappears, showing this first activity in the list.

13. Once again, in the **Activities** group, click **New**, and complete the details for a merge activity:

a. Add a **Name** for the activity.

Suggested value: `Merge`.

b. Ensure that the **Continue on Error** check box is clear.

c. Set the **Sequence** number to a value higher than the identification step.

Suggested value: `600`.

d. From the list of **Source Datasets**, select only the one identified in the previous activity.

The suggested value was `BMC.FNMS`.

e. From the list under **Target Dataset**, select the `BMC Asset` dataset.

f. In the **Precedence** group, click the **+** icon to add a new **Precedence Association Set**.

Suggested value: `BMC.FNMS,BMC Asset`.

g. Click **Done** to save details of this activity.

The **Activities** group reappears, showing your two activities.

14. In the **Schedule** group, click **New**.

    Set the values to suit your business processes, with a suggested frequency being once every 2-4 weeks.

15. At the bottom of the **Job Editor**, click **Save** to store your job details in the repository.

Tuning Creation of Configuration Items

Atrium requires an initial run to populate the database, and then customization for regular scheduled operations. In the first run of the adapter, all the products and computers exported from FlexNet Manager Suite are inserted as configuration items in the CMDB. Thereafter, matching records should be updated, and only new items inserted. We therefore run the first pass with the default settings, and then modify the settings to schedule the regular data exchange.

To configure the creation process for configuration items:

1. In **BMC Atrium Integrator** console, from the list of **Jobs** on the left, choose the `BMCAtriumAdapter` job.
2. In the toolbar, click the third icon (with the play button) to run the selected job once.

As there may be a large backlog of data to handle, and Spoon processes a line at a time, the first import can be quite time-consuming. Wait for the first import to complete successfully before continuing with this procedure.

3. In Atrium Integrator, update the way Atrium receives the export of computers as follows:
   a. Open the `ExportComputersTransform` transformation from the repository.
   b. Double-click the `CMDBoutput` step to expose its properties.
   c. Clear (turn off) the **Always Insert CIs** check box.
   d. Select (turn on) the **Only insert new CIs** check box.
   e. Click **OK** to close the `CMDBoutput` step, and click **Save** to store the changes to the transform.
4. Update the processing of the export of products:
   a. Open the `ExportProductsTransform` transformation from the repository.
   b. Double-click the `CMDBoutput` step to expose its properties.
   c. Clear (turn off) the **Always Insert CIs** check box.
   d. Click **OK** to close the `CMDBoutput` step, and click **Save** to store the changes to the transform.

Scheduling the Adapter

The adapter to exchange data between FlexNet Manager Suite and BMC Atrium is driven by a schedule set up in Atrium.

To configure the schedule for data exchange:

1. If you are not already in the Atrium Integrator console:
   a. Login to BMC Remedy Mid-Tier using user name and password provided by BMC Administrator.
   b. Click the **Applications** tab on the left edge.

The **Applications** menu expands.

c. Select **Atrium Integrator**, and then click **Atrium Integrator Console**.

2. In the **Jobs** list on the left, select your `BMCAtriumAdapter` job.

3. In the tool bar, select the icon to manage the job schedule ( ).

4. Select **Active**.

5. Select the **Create new schedule** option.

The **Schedules - "BMCAtriumAdapter"** page appears.

6. Complete the details for the regular operation of the adapter (this schedule controls data flow both ways, for import and export):

a. Give the schedule a meaningful name in the top right corner (**Enter Schedule Name**).

b. Use the **Start Time** spinners to dial up the time of day to start operation of the adapter.

Tip: *This is local time on the BMC Remedy AR System server.*

c. In the **Recurrence Type** area, specify how frequently you want the job to run.

Recommendation: The process is quite time-consuming. In a stable environment, consider a monthly data exchange (**Interval Based**).

d. Depending on your choice, complete the **Recurrence Details**.

e. Click **Save**.

The integration between FlexNet Manager Suite and BMC Atrium is now operational.
However, before the next scheduled run, you should consider any further customization that may be necessary, especially in relation to large data sets.

Verifying Data Export to BMC Atrium

This process confirms that data has been exported from FlexNet Manager Suite and successfully imported by Atrium. This procedure uses the Atrium Core application (not Atrium Integrator).

To verify successful data transfer into Atrium:

1. If you are not already in the Atrium Core console:

   a. Login to BMC Remedy Mid-Tier using user name and password provided by BMC Administrator.

   b. Click the **Applications** tab on the left edge.

   The **Applications** menu expands.

   c. Select **Atrium Core**, and then click **Atrium Core Console**.

   A new window opens with the **Application Launcher** tab on the left edge.

2. Click the **Application Launcher** tab, and in the fly-out menu, click **Applications**.

3. In the sub-menu, click **Explorer**.

   The **Explorer** tab opens, with multiple accordion folds. Ensure that the **Find** fold is open.

4. In the **Find** area, use the drop-down list on the left below the tool icons to select the `BMC.FNMS` dataset.
5. Optionally choose an object type to search for, or expand the **Quick search of all CIs**.
6. Optionally, add data (or partial values) to the **Name** and **Short Description** fields (and controls for matching) to narrow the search for particular configuration items, or leave the fields blank for a full listing of all results.
7. Click **Search**.

   A list of results (shown inset) displays. Validate that the data is as expected.

3

Additional Customization of the Adapter

The following notes help with minor changes to configuration, especially for future changes to configuration. The one case that may be critical to your initial implementation is the first topic, covering preparations for large datasets.

Preparing for Large Datasets

Large numbers of data records require a customized environment variable. If your integration between FlexNet Manager Suite and BMC Atrium will exchange large numbers of records (for example, 100,000 computer records or more), you need to configure an environment variable on the Atrium server.

To prepare for large datasets:

1. Ensuring that you are logged onto the Atrium server with Administrator privileges, open the Windows start menu, and right-click **Computer**.
2. From the context menu, select **Properties**.

   The Control Panel page of system properties appears.
3. In the navigation bar, select **Advanced system settings**.

   The **System Properties** dialog appears. Confirm that the **Advanced** tab is selected.
4. Click **Environment Variables...**.

   The **Environment Variables** dialog appears.
5. In the **System Variables** (lower) group, click **New...**.

   The **New System Variable** dialog appears.
6. Insert the **Variable name**:

   ```
   _JAVA_OPTIONS
   ```

7. Insert the **Variable value**:

```
-Xms512m -Xmx15360m
```

   This configures the Java heap size above 15GB.

8. Click **OK** three times to close the various dialogs, and you can close the Control Panel window.

Updating Connection Details

When system passwords change, you need to update the adapter's configuration file. Connection details are stored, encrypted, in the adapter's configuration file (by default located at `C:\Program Files\Flexera\BMCAtriumAdapter\ FNMPExports\Utility\ FNMPDataExportUtility.exe.config` on the Atrium server). The encryption process happens the first time the adapter runs after the configuration file is edited. Once the process has run, the connection strings are unreadable and not editable, something similar to the following:


```
<connectionStrings configProtectionProvider="DataProtectionConfigurationProvider">
    <EncryptedData>
      <CipherData>
        <CipherValue>AQAAANCMnd8BFdERjHoAwE/Cl+sBAAAAeNhcGMAVK0uVGNTq
Og/WbQQAAAACAAAAAAADZgAAwAAAABAAAACTO9kpn6BptpLvsXExg1UBAAAAAASAAACgA
AAAEAAAAELCyiwz5AXZw9xZXfEPiAAAAgAAPJQYe+G9AfScFMJTYgA0NDbAgZdRA9nB91
DN42A1xjeCskUs9+KNjVU1PSFRV4ujta40evf3IOZy5odyHsIrJRCKOOGdhDb1wh4ISEk
pJk/QDna6LeCbbtXXsQK2LoAHQc/plz77UkQZxnxkL5ElPIGHl6AojEXT2F5NGjElJX6G
XbUXQDkNnDfi2o6XI/CDbX8gCuMonY1cTLYGe6+AQPpDgcY3rA02ZFOs7/Zb0cKOw7IoZ
dB6H8OIvrClSzqNkVBd3YfLhP/KOrkQFP8orqj54BJW74E1v3VUZnte1ESgLA5MYb/F9A
h3M5xi2Q6ITXOQmVRGESrivsdqr6nyGz5APx2yVBuEcoVhpOMYURbEbBSW+6/aydg8nY1
DrcMzkPlXiZ0CQs4yZYHSWt3+bFEN30xh6XKFH7Tpc8e5y9Tq1Bhwk+Kw6AFfZhcdewAp
J4ZkGAr4ixE6gNqWXnomk2puKNFImhJfqLLIuQx9T00Uu2bjJ9Y+dUlrcuov12hQXOCh9
nqOdmeIQHPXgw8//eHYOzwy2TgMcq2M2LxXRbQqCTeWtlP1ucJVyct9gnJlk2L3FSBNgM
u1C9mncR7MsGDBWoQMXnwC5+Y6P1GT45sPOedBQvIQITj7MCuzfgDD1R9c7w1gejTGmrl
3Kmf33tGPMRYPV7CPNET3DUV9AGQUAAAAaIEkjyqfExrbeiYJvG2usqYO3Nk=</CipherValue>
      </CipherData>
    </EncryptedData>
  </connectionStrings>
```


To replace connection strings and/or passwords:

1. Open the adapter's configuration file in a flat text (or XML) editor.

   Default location: `C:\Program Files\Flexera\BMCAtriumAdapter\ FNMPExports\Utility\ FNMPDataExportUtility.exe.config`

2. Locate the `connectionStrings` element (similar to that shown above), select it all, and delete it.

3. Copy the following example, paste into the same place in the configuration file, and replace the placeholder values (including their containing brackets) as described below (you may also remove the white space from the connection strings):

```
<connectionStrings>
    <clear />
          <add name="FNMPDBConnection"
              connectionString="Data Source=[FNMPDBServerName];
                                Initial Catalog=FNMSCompliance;
                                User ID=[accountName1];Password=[somePassword];
                                Integrated Security=false;
                                Connection Timeout=60;"/>
          <add name="IntermediateDBConn"
              connectionString="Data Source=[intermediateDBServerName];
                                Initial Catalog=IntermediateDB;
                                User ID=[accountName2];
                                Password=[anotherPassword];
                                Integrated Security=false;
                                Connection Timeout=60;"/>
  </connectionStrings>
```

where:

Placeholder	Notes
[FNMPDBServerName]	The database server (and optionally, the database instance name) containing the FlexNet Manager Suite compliance database. You may use: • A dot (.) when the database is installed on the same server where the adapter executable runs • The flat name of the server • The IP address of the server • Where the database instance is not the default instance on that server, any of the above with a slash (/) separator, and the database instance name.
FNMSCompliance	This is the recommended database name for the compliance database. If you used a different name, substitute the name you used.
[accountName1]	The user name that the executable uses to connect to the compliance database.
[somePassword]	The (newly current) password for the above account.
[intermediate DBServerName]	The database server (and if necessary, instance) for the intermediate database. The same formats are supported as described above for [FNMPDBServerName].
[accountName2]	The user name that the executable uses to connect to the intermediate database.
[anotherPassword]	The password for [accountName2].

4. Save the edited configuration file.

   The file is saved in plain text. The next time the adapter runs, this section of the configuration file is automatically encrypted. If you are unhappy about leaving the file unencrypted until then, trigger an additional run of the

adapter by adding a one-time schedule (see Scheduling the Adapter).

Updating Email Details

With data flows in both directions between FlexNet Manager Suite and BMC Atrium, errors may occur in either export or import. The adapter sends emails to inform you when an error occurs. The emails for the two directions are defined separately:

- For export from FlexNet Manager Suite, the data export utility generates the emails, with settings in its configuration file. If the details of your email server are changed, you need to update the adapter's configuration file. It also defines who receives the alerts, and the address from which alerts originate. You can modify any of those detail with this procedure.
- Similarly, for imports from BMC, the email is generated by Atrium based on settings in that system. Those settings are also summarized in this procedure, since it is likely that a change in your infrastructure or personnel affects both kinds of email alerts equally.

We start with emails for exports from FlexNet Manager Suite; and then continue with those for imports to the same system (that is, data is exported from BMC Atrium).

To update email details:

1. Open the adapter's configuration file in a flat text (or XML) editor.

   Default location: `C:\Program Files\Flexera\BMCAtriumAdapter\ FNMPExports\Utility\ FNMPDataExportUtility.exe.config`

2. Locate the `<appender>` element for the `SmtpAppender` node in the file and update the settings marked here with placeholders:

```
<appender name="SmtpAppender" type="log4net.Appender.SmtpAppender">
        <to value="toaddress@somedomain.com"></to>
        <from value="fromaddress@somedomain.com"></from>
        <subject value=" Atrium Integration Adapter Error"></subject>
        <smtpHost value="smtp.somedomain.com"></smtpHost>
```

   where:

Value	Notes
toaddress @somedomain.com	Valid email address for the person who is to receive alerts when errors occur during export.
fromaddress @somedomain.com	Valid email address for an account known on the specified SMTP host that is identified as the sender of the email when an error occurs.
smtp.somedomain.com	The fully qualified domain name of the SMTP email server.

3. Save the edited file.
4. Open the Atrium Integrator Spoon tool.

5. Select **File > Open...**, and select `BMCAtriumAdapter.kjb` from the repository.

   A map of the job displays.

6. To configure email alerts for errors occurring on the Atrium side of the adapter:

   a. In the `BMCAtriumAdapter` job, right-click on the `Mail` step in the center.

   b. From the option menu, select **Edit job entry**.

      The **Job mail details** dialog appears.

   c. In the **Addresses** tab, insert the following:

      - **Destination address** — the person who is to receive the email alert for each error.
      - **Sender name** — the value to appear on the emails as the sender.
      - **Sender address** — The email address from which the email will appear to come. This must be a valid email account known to the mail server (specified next).

   d. Switch to the **Server** tab, and complete the **SMTP Server** details.

   e. Click **OK**.

The revised email settings for data transfer in both directions take effect for the next run of the adapter.

Turning Off Email Alerts for Export Errors

You can disable email alerts for problems arising during export from FlexNet Manager Suite. This procedure provides a quick way to temporarily disable emails sent by the data utility during data export from FlexNet Manager Suite for transfer to BMC Atrium. This procedure turns the emails off without removing the settings details.

Tip: *For emails generated by BMC Atrium about data preparation for import to FlexNet Manager Suite, there is no single "off switch". To stop those emails, re-do the settings to remove the email details (as described in Updating Email Details).*

To turn off email alerts:

1. Open the adapter's configuration file in a flat text (or XML) editor.

   Default location: `C:\Program Files\Flexera\BMCAtriumAdapter\ FNMPExports\Utility\ FNMPDataExportUtility.exe.config`

2. Locate the following section (towards the end of the configuration file):

```
<root>
  <level value="ALL" />
  <appender-ref ref="RollingFileAppender" />
  <appender-ref ref="SmtpAppender" />
</root>
```

3. Comment out the `appender-ref` element for the `SmtpAppender`:

   XML comments are surrounded by `<!-- -->`:

```
<root>
  <level value="ALL" />
  <appender-ref ref="RollingFileAppender" />
  <!-- appender-ref ref="SmtpAppender" / -->
</root>
```

4. Save the amended file.

Email alerts for export errors are suspended until you reverse this process, re-editing the file to remove the comment tags and restored the original value.

Changing the Log File

You can rename or relocate the log file generated by the adapter for exports from FlexNet Manager Suite.

By default, five copies of the log file, each capped at 10MB, rotate in the nominated folder (total disk space requirement is therefore about 50MB for logging). You can customize the location and base name of the log file as follows:

To rename or relocate the log files:

1. Open the adapter's configuration file in a flat text (or XML) editor.

   Default location: `C:\Program Files\Flexera\BMCAtriumAdapter\ FNMPExports\Utility\ FNMPDataExportUtility.exe.config`

2. Locate the following section in the file:

```
    <appender name="RollingFileAppender"
type="log4net.Appender.RollingFileAppender">
      <file value="C:\Program Files\Flexera\BMC Atrium Adapter\Log.txt" />
      ...
    </appender>
```

3. Edit the `value` attribute with the new path and file name.

   Tip: *Keep in mind that the file name will be expanded with dating information to differentiate the various logs.*

4. Save the modified file.

4

Appendices

The following topics provide data mapping between attributes recorded in the FlexNet Manager Suite database and those recorded in the BMC Atrium CMDB. In addition, some values are generated as part of the export and import processes, and are included in the following listings.

Two separate data objects are exported from FlexNet Manager Suite: `Computers`, and `Products`. A much smaller dataset is imported from Atrium into FlexNet Manager Suite, updating information about assets to show their role and owner.

Appendix 1: Export of Computers

For optimal performance, exports are differential (that is, only data changed since the last export is included). The following table shows:

- Columns from the `ComplianceComputers` table (and related tables) in FlexNet Manager Suite that are exported
- The equivalent column name in the `ComputerSystemChanges` table of the intermediate database (in case you wish to inspect the dataset there)
- The matching column in the `BMC_ComputerSystem` table in the Atrium CMDB.

Note:

1. *Some values are not read from the source database, but inserted as fixed values. These are shown in double quotation marks, and explained in the associated comments.*
2. *Some values exported from FlexNet Manager Suite to the intermediate database are not transferred to BMC Atrium. Instead, they are used to manage the exported data, removing records not relevant to the CMDB and so on.*

Compliance Computer	ComputerSystem Changes	BMC_ ComputerSystem	Comments
Not applicable.	*GeneratedTokenID*	TokenID	TokenID takes different formats for different types of computers. In the following examples, the placeholder *X* represents any hexadecimal digit (0-9, A-F). • For physical hosts and computers, the TokenID is *hostname*:*DNSdomainName*. • For some virtual machines, TokenID takes the form of a prefix, a colon, and a UUID: ◦ For VMware, VI-UUID:*XXXX-XX-XX-XX-XXXXXX* ◦ For Hyper-V, HYPERV-ID:*XXXX-XX-XX-XX-XXXXXX* ***Note:*** *With Hyper-V, the UUID is only available on the physical machine, so TokenId is only set for virtual machines that have been successfully linked to their hosting physical machines.* ◦ For Xen (including Oracle VM), XEN-ID:*XXXX-XX-XX-XX-XXXXXX* ◦ For KVM (including RedHat Enterprise Virtualization), KVM-ID:*XXXX-XX-XX-XX-XXXXXX*. TokenID is generated dynamically, checking the value in ComputerType. For virtual machines, the value in ComputerToVirtualMachine_VMTypeID is used to determine the prefix, and the UUID is attached from ComputerToVirtualMachine_UUID.
Compliance ComputerID	ComputerID	CITag	The identifier for the computer in the FlexNet Manager Suite database, passed to BMC as the configuration item tag. (This is passed back again if Asset records are imported by FlexNet Manager Suite.)

Compliance Computer	ComputerSystem Changes	BMC_ ComputerSystem	Comments
ComputerName	ComputerName	HostName	The name of the computer, limited to 256 characters.
	“DNS”	Name Format	A fixed string value determining the data type for Name.
[ComplianceDomain] QualifiedName	Domain	Domain	The fully qualified domain name for the domain where this computer exists. Extracted from ComplianceDomain.QualifiedName using ComplianceComputers. ComplianceDomainID as a foreign key. Used when there is no value for FlatDomain received from FlexNet Manager Suite.
[ComplianceDomain] FlatName	FlatDomain	Domain	The unqualified name of the domain containing the computer. Extracted from ComplianceDomain.FlatName using ComplianceComputers. ComplianceDomainID as a foreign key. Where this value is present, it is stored in BMC_ComputerSystem.Domain; if not, the qualified domain name is used instead.

Compliance Computer	ComputerSystem Changes	BMC_ ComputerSystem	Comments
[Compliance ComputerStatus] DefaultValue	ComputerStatus	(Not stored.)	The current status of this computer within the inventory records. Values include: • New (this is the first appearance of this computer in inventory) • Ignored (an operator has marked this computer to be ignored) • Registered (this inventory record is linked to an asset record) • Awaiting Inventory (a dummy computer record, allowing its associated asset record to exist, which will be removed when an operator links the asset record to a real inventory item) • Discarded. These values are extracted from ComplianceComputerStatus. DefaultValue based on the foreign key ComplianceComputers. ComplianceComputerStatusID. ***Tip:*** *The DefaultValue is the US English representation of the values, not localized.* The status is used to make sure that computers that are Awaiting Inventory, Discarded, or Ignored are not transferred to BMC Atrium.
Not applicable.	"Workstation"	CapabilityList Primary Capability	Fixed literal string inserted in the BMC database.

Compliance Computer	ComputerSystem Changes	BMC_ ComputerSystem	Comments
[Compliance ComputerType] DefaultValue	ComputerType	isVirtual	The boolean isVirtual is set true whenever the ComputerType in the intermediate database is either VMHost or Virtual Machine. Possible values are extracted from ComplianceComputerType. DefaultValue based on the foreign key ComplianceComputers. ComplianceComputerTypeID .
InventoryDate	InventoryDate	(Not stored.)	The date inventory was last collected from this computer.
Manufacturer	Manufacturer	ManufacturerName	Values from FlexNet Manager Suite are provided to avoid data normalization by the CMDB that may modify names.
SerialNo	SerialNo	SerialNumber	The serial number of the computer.
[VirtualMachine] Compliance ComputerID	ComputerTo VirtualMachine_ Compliance ComputerID	(Not stored.)	Populated only for virtual machines. A foreign key that links records in the VirtualMachine table to the matching records in the ComputerCompliance table. Used here in the calculation of TokenID.
[VirtualMachine] UUID	ComputerTo VirtualMachine_ UUID	(Not stored.)	Populated only for virtual machines. The UUID (Universally Unique Identifier) of the virtual machine. Used to match virtual machine properties to their associated ComplianceComputer. Used here in the calculation of TokenID for virtual machines.

Compliance Computer	ComputerSystem Changes	BMC_ ComputerSystem	Comments
[VirtualMachine] VMTypeID	ComputerToVirtual Machine_ VMTypeID	(Not stored.)	Populated only for virtual machines. A foreign key to the VMType table, which identifies the kind of virtual machine, including: • VMware • Hyper-V • LPAR • WPAR • nPar • vPar • SRP • Zone • Unknown. Used here in the calculation of TokenID.
Not applicable.	UpdatedDate	(Not stored.)	Used when identifying changed data for differential exports to BMC Atrium.
Compliance Computer TypeID	Compliance Computer TypeID	(Not stored.)	This value is used to filter out remote devices (those from which inventory cannot be collected) and VDI templates, which are not relevant to the CMDB.
Not applicable.	ViewID	(Not stored.)	Deprecated and not in use.
Not applicable.	TenantID	(Ignored.)	Always contains a zero value (a string of 16 zeroes) for an on-premises implementation.
Not applicable.	“BMC_ COMPUTERSYSTEM”	ClassID	Fixed literal string inserted in these records to identify the data class in the data set.
Not applicable.	"*CompanyName*"	Company	A fixed value that must match the company identifier embedded in the Atrium CMDB.
Not applicable.	"*DataSetID*"	DataSetID	Value attached to the dataset considered. Must match the value embedded in the Atrium CMDB.

Appendix 2: Export of Applications/ Products

The term 'product' is used in two slightly different ways in FlexNet Manager Suite and BMC Atrium.

- In FlexNet Manager Suite, the base unit of software is called an "application", which is specific to a particular release (version) and edition. A 'product' is the same software unit, but across multiple versions (and perhaps editions as well). In our terms, then, FlexNet Manager Suite exports application records. To confuse things further, the underlying database table is called `SoftwareTitle`.
- In Atrium, a "product" is the base unit of software, and related to an individual installation. The CMDB also offers a catalog of the products available for installation. In BMC terminology, Atrium imports product records, from which the Product Catalog may also be updated.

The following table shows:

- Columns from the `SoftwareTitle` table (and related tables) in FlexNet Manager Suite that are exported
- The equivalent column name in the `ProductChanges` table of the intermediate database (in case you wish to inspect the dataset there)
- The matching column in the `BMC_Product` in the Atrium CMDB. This indirectly populates the Product Catalog.

Note:

1. *Some values are not read from the source database, but inserted as fixed values. These are shown in double quotation marks, and explained in the associated comments.*
2. *Some values exported from FlexNet Manager Suite to the intermediate database are not transferred to BMC Atrium. Instead, they are used to manage the exported data, removing records not relevant to the CMDB and so on.*

SoftwareTitle	ProductChanges	BMC_Product	Comments
Not applicable.	*GeneratedTokenID*	ParentCITag	The token ID generated for the computer system on which this product is installed is assigned to the `ParentCITag` column in BMC. For details about the generated `TokenID`, see Appendix 1: Export of Computers.
SoftwareTitleID	ApplicationID	(Not stored.)	Used internally in data management.

SoftwareTitle	ProductChanges	BMC_Product	Comments
FullName	ApplicationName	ShortDescription	By default, the full name of the application is the concatenation of the product, version, and edition fields (for example, “Microsoft Office Professional 2010”). However, an operator may overwrite this with any preferred value.
[SoftwareTitleVersion] VersionName	ApplicationVersion	MarketVersion	The marketing version number (such as "2010"). Extracted from SoftwareTitleVersion.VersionName using SoftwareTitle.SoftwareTitleVersionID as a foreign key.
(Generated ID)	FlexeraID	TokenID	FlexeraID is a unique identifier for application records used for unambiguous identification across various Flexera solutions. The format is either: • For applications published in the FlexNet ARL, arl://MGS-APP-00000000000 (where the 11-digit number varies by application) • For applications defined locally, app://*serverGUID*/LOCAL-APP-*SoftwareTitleID* (where *serverGUID* depends on the identity of the central application server, and *SoftwareTitleID* is the value of SoftwareTitleID assigned to this application). The result is presented as the TokenID in the BMC_Product table, and as SignatureID in the Product Catalog.

SoftwareTitle	ProductChanges	BMC_Product	Comments
[SoftwareTitle Publisher] PublisherName	Publisher	ManufacturerName	Identifies the company that produces the application/product (for example, “Microsoft”). Extracted from SoftwareTitlePublisher. PublisherName via SoftwareTitleProduct. SoftwareTitlePublisherID, using SoftwareTitle. SoftwareTitleProductID as a foreign key.
[Compliance Computer] ComputerName	ApplicationTo Installation_ ComputerName	SystemName	Name of the computing device reported in inventory (such as “vincent-ltp”). Extracted from ComplianceComputer. ComputerName using the foreign key InstalledSoftware. ComplianceComputerID. The view InstalledSoftware is itself linked to the application by the foreign key SoftwareTitleID.
ComputerID	InstallationTo Computer_ ComputerID	(Not stored.)	The computer identifier, extracted in much the same way as described for ComputerName. Used internally for data management.
[Compliance Connection] LastImportDate	InstallationTo Computer_ InventoryDate	(Not stored.)	The date of the most recent inventory collection that recorded the installation of this application on the current computer. When no other value is available, this defaults to the date of data export from FlexNet Manager Suite. Used internally for data management.
CategoryID	InstallationTo Application_ Category	Category	First level of application categorization. The full path of categorization is extracted from the GroupEx table using SoftwareTitle. CategoryID as the foreign key. The full path is then broken up on its / separators, and the first level is passed to Category in the BMC_Product table.

SoftwareTitle	ProductChanges	BMC_Product	Comments
CategoryID	InstallationTo Application_ Category	Type	Second level of application categorization. (See Category for details.)
CategoryID	InstallationTo Application_ Category	Item	Third level of application categorization. (See Category for details.)
Not applicable.	ViewID	(Not stored.)	Deprecated and not in use.
	TenantID	(Ignored.)	Always contains a zero value (a string of 16 zeroes) for an on-premises implementation.
Not applicable.	EvidenceVersion		Deprecated and not in use.
Not applicable.	“BMC_PRODUCT”	ClassID	A string literal that identifies the class in the BMC dataset.
Not applicable.	"*CompanyName*"	Company	A fixed value that must match the company identifier embedded in the Atrium CMDB.
Not applicable.	"*DataSetID*"	DataSetID	Value attached to the dataset considered. Must match the value embedded in the Atrium CMDB.
Not applicable.	*FullName*:*VersionName*	Name	Full name separated by a colon from the market version of the software product (for example, “Microsoft Office Professional:2010”).
Not applicable.	“ProductName: Version“	NameFormat	A string literal describing the format of the Name attribute.
Not applicable.	*FullName* on *ComputerName*	Description	Composite of the software name and device name (such as “Microsoft Office Professional 2010 on vincent-ltp”).
Not applicable.	“Deployed”	Status	A fixed value, since the records are extracted from the installed software listings.
Not applicable.	*Calculated value*	MarkAsDeleted	Boolean (0 = no, 1 = yes). Whether or not this product has been deleted from the computer system under consideration. Note that this flag is raised only for the first import after the product is deleted; thereafter, further imports do not contain this row.

Appendix 3: Import of Assets

The following table shows:

- Columns from the BMC_ComputerSystem view (and AST:ComputerSystem form) in Atrium CMDB that are exported
- The equivalent column name in the Assets table of the intermediate database (in case you wish to inspect the dataset there)
- The matching columns in the FlexNet Manager Suite database where this information is imported.

BMC_ComputerSystem	Assets	ComplianceComputer	Comments
Reconciliation Identity	Reconciliation Identity	Not stored.	Used internally to link the BMC_ComputerSystem view and AST:ComputerSystem form.
Domain	DOMAIN	[ComplianceDomain] QualifiedName and FlatName	The domain extracted from BMC Atrium is processed to provide a dot-separated qualified name in the form flexerasoftware.com. The same value is processed again to provide a flat domain name (such as flexerasoftware). Where these values match existing entries in the ComplianceDomain table, no further action is taken with them (other than to return the ComplianceDomainID key); but where they are not already known, they are added to this table for use in FlexNet Manager Suite. As well, the ComplianceDomainID value returned form this update is then written into the ComplianceComputer record, which may update the domain for the current computer.
HostName	HostName	ComputerName	The name of this computer. Based on a matching ComplianceComputerID, this value may be over-written with the value imported from Atrium.

BMC_ComputerSystem	Assets	ComplianceComputer	Comments
CITag	CITag	Compliance ComputerID	The identity for the computer record. Where this matches an existing record, its other columns may be updated. Otherwise, a new record is created for ComplianceComputer in FlexNet Manager Suite.
	InstanceId	Not stored.	Deprecated and not in use.
ImportLog table	LastImportDate	Not stored.	Calculated from data in the ImportLog table.
[AST: ComputerSystem] System Role	System Role	Custom property Role in the **Details** tab (**User** group) of the inventoried computer properties.	Both systems record roles for servers, but the FlexNet Manager Suite role has various licensing implications, and so is not updated by imports from Atrium. Instead, a read-only custom field is added to FlexNet Manager Suite to accept the asset role for each BMC_ComputerSystem. The role is displayed in the **Details** tab of the computer properties. Values include: • Lab • Development • SIT • UAT • Training • Performance • Contingency • Production.
[AST: ComputerSystem] Reconciliation Identity	Reconciliation Identity	Not stored.	Used internally to link the BMC_ComputerSystem view and AST:ComputerSystem form.
[AST: ComputerSystem] Assigned To	Assigned To	AssignedUserID	The ID associated with ComplianceUser.UserName, written here to link the ComplianceComputer to the user record.

BMC_ComputerSystem	Assets	ComplianceComputer	Comments
	Modified Date	Not stored.	Deprecated and not in use.
[AST: ComputerSystem] Owner_name	Owner_name	[ComplianceUser] UserName	The full name of the user assigned to this computer. FlexNet Manager Suite defaults to a concatenation of title, first name, middle name, last name and suffix, so any of these parts may be present, based on the record in BMC Atrium. Where this matches an existing user record, no further action is taken with it; otherwise a new user record is created. In either case, the ID for this user is returned, and written into the AssignedUserID column. ***Tip:*** *Beware of small changes in record data from Atrium causing near-duplicate records to represent the same person. Also note that when new records are created, you may wish to update additional details about this person in FlexNet Manager Suite.*
[AST: ComputerSystem] AssetLifecycle Status	AssetLifecycle Status	Compliance Computer StatusID	A translation of the BMC Atrium status values into the nearest equivalent status value supported in FlexNet Manager Suite. This is then mapped by the Business Importer into the correct ID for the status value before being written into the ComplianceComputer record.
[AST: ComputerSystem] DNS Host Name	DNS Host Name	Not stored.	Host name from the BMC_ComputerSystem is used in preference to this one.
[AST: ComputerSystem] Domain	Domain_1	Not stored.	Domain from the BMC_ComputerSystem is used in preference to this one.
[AST: ComputerSystem] Instance Id	Instance Id	Not stored.	Used internally.

BMC_ComputerSystem	Assets	ComplianceComputer	Comments
[AST: ComputerSystem] Data Set Id	Data Set Id	Not stored.	Used internally.

IV

AWS EC2 Connector

This part introduces the AWS EC2 connector, with a particular focus on managing licenses that you provide for software used in the cloud (often called "bring your own software license", or BYOSL). The complementary case, where you purchase instances fully provisioned by AWS (including the provision of relevant licenses), is of less interest to your license management team, and is perhaps more the focus of your finance team.

The AWS EC2 connector is only a part of a complete BYOSL management strategy for *instances* (typically, virtual machines that are tracked as inventory devices) that are hosted for you by this cloud service provider. By itself, the AWS EC2 connector does not gather sufficient data to allow for license management: for example, it does not gather software inventory for the applications that are running on your cloud-hosted devices.

For that reason, this part covers more than the AWS EC2 connector itself. As well as details of prerequisites, set up, and data gathering with the AWS EC2 connector, the part includes some insights into:

- Processes for gathering *complete* hardware and software inventory for your cloud-based instances
- Considerations for license management
- Preparing different Amazon Machine Images (AMI) from which to launch instances capable of reporting software and hardware inventory.

In fact, using those guidelines, you *could* configure inventory gathering in AWS EC2 without using the AWS EC2 connector at all. Provided that you are deploying the latest FlexNet inventory agent (13.2.x or later), inventory alone even returns the cloud service provider. What the AWS EC2 connector adds is:

- Automatically setting the **Hosted in** property in the inventory device records created when inventory is returned
- Adding the cloud service provider and instance details when these are missing from inventory collected by legacy versions of FlexNet inventory agent, or from third-party inventory sources
- Populating the **Cloud Service Provider Inventory** page with records of instances currently running in your AWS EC2 environment, including a few properties that are additional to the normal hardware and software inventory
- Importing permanent records of instances that were previously running but that have now been terminated (and for which, as a result, any prior inventory devices records have now been deleted)
- Triggering automatic deletion (completed at the next full import and compliance calculation) of any inventory device record linked to an instance that is now terminated.

This part focuses on background about the AWS EC2 connector, and additional material you need in order to build license

management around the AWS EC2 connector. You will not find details about how to choose the right connection method for your business processes (there are three method available), or how to undertake the actual connection here. Instead, step-by-step instructions for configuring the connection to AWS ECS are available on the online help at ***FlexNet Manager Suite Online Help > Inventory Beacon Help > Inventory Systems Page > Connecting to External Inventory Systems > Managing PowerShell Connections > Managing AWS EC2 Connections***.

1

Prerequisites and Setting Up

The AWS EC2 connector is configured on a convenient inventory beacon, for which there are the following prerequisites:

- PowerShell 3.0 or later is running on Windows Server 2008 R2 SP1 or later, or Windows 7 SP1 or later; with the PowerShell execution policy set to `RemoteSigned`.
- The FlexNet Beacon software installed on the inventory beacon must be release 13.1.1 (shipped with FlexNet Manager Suite 2018 R2) or later.
- A web browser is installed and enabled on the inventory beacon.
- You must log onto the inventory beacon, and run FlexNet Beacon, using an account with administrator privileges.
- You must have downloaded AWS Tools for Windows PowerShell from https://aws.amazon.com/powershell/, and installed them on the inventory beacon. The minimum required version of these tools is 3.3.283.0.

Tip: *To check the version installed on your inventory beacon:*

1. *As administrator, run AWS Tools for Windows PowerShell.*
2. *Execute the `Get-AWSPowerShellVersion` cmdlet.*

New versions are available for download from https://aws.amazon.com/powershell/.

Note: *The permissible values for* ***Instance region*** *are currently hard coded in the AWS Tools for Windows PowerShell. This means that if AWS create additional regions, and you want to have instances in one of the new regions, you will need to update AWS Tools for Windows PowerShell at that time.*

On the AWS side, you must first create:

- Within AWS, you must create a policy allowing access to your EC2 service
- Similarly, you must create a policy allowing access to an Identity and Access Management (IAM) entity
- The IAM roles to grant access to AWS resources (not required when using the Configuring Connections to AWS EC2 using IAM Users method)
- The IAM user account (still within AWS) with minimum privileges that makes the connection to AWS APIs and imports the available data (Configuring Connections to AWS EC2 using IAM Roles ***without*** FlexNet Beacon installed on EC2 instance only)

- Finally, on the inventory beacon that is to make the connection to AWS, you must specify the connection (which is automatically scheduled for you).

Using Proxy Connections

The connection to AWS supports optional use of a proxy for connections that *do not* connect using a locally installed FlexNet Beacon.

Important: *A proxy server cannot be specified when connecting using a locally installed beacon in AWS using role-based access. This is because the URL which is accessed is a local address of 169.254.169.254 which cannot and will not be routed through proxies.*

Step-by-step instructions for creating these, and then configuring the connection to AWS ECS, are available on the online help at ***FlexNet Manager Suite Online Help > Inventory Beacon Help > Inventory Systems Page > Connecting to External Inventory Systems > Managing PowerShell Connections > Managing AWS EC2 Connections***.

2

Thinking about Inventory and Licensing

In many ways, collecting inventory and working out license consumption for instances in AWS EC2 is much the same as licensing virtual machines on host servers sitting within the datacenter in your own offices. However, there are some differences that are worth thinking about. For example, within your own datacenter, you have full control of, and presumably full inventory for, the host server as well as the VMs running on it; and for some license types, host details are relevant even when licensing software on a virtual machine. Depending on what you purchase, this may not be the case in AWS EC2.

The following topics bring together some points to consider, first about inventory collection, and then about licensing.

Collecting Inventory from Instances

The AWS EC2 connector gathers useful information, but it cannot gather information valuable for license consumption calculations from each of the instances running in your AWS cloud. To gather useful inventory from all instances, best practice is to include the FlexNet inventory agent (release 13.2.`buildNumber` or later) in the Amazon Machine Image (AMI) from which your devices are instantiated. For details, see the AWS documentation, *Create a Standard Amazon Machine Image Using Sysprep* for Windows, or for Linux *Creating an Amazon EBS-Backed Linux AMI*. For additional guidance about the kinds of customizations your AMI may need (depending on the life expectancy of instances to be launched from the image), see either:

- Configuring an AMI for Short-Lived Instances
- Configuring an AMI for Longer-Life Instances.

Unique device naming is critical

It is critical to your license management in AWS to ensure that each instance is *uniquely* named on instantiation. If you do not do this, incoming inventory reports are not differentiated, and are interpreted as updated inventory from a single device, named with the default device name provided by the base image.

One easy way that *appears* to give each instance a unique name different from its AMI is through the AWS **Ec2 Launch Settings** dialog, by selecting the **Set Computer Name** check box (or setting the Boolean in the `LaunchConfig.json` file). However, be aware that, since this name is of the form `ip-hexInternalIP`, the name may not be unique *over*

time, particularly if you have short-life instances. If an internal IP address is dropped when a first instance is shut down, and reused for *another* instance that is instantiated later, then these different instances are overlaid in the FlexNet *inventory* database as being updates to the same device (having the same name). However, when the inventory records are then imported over time into the compliance database, they are differentiated by their different instance IDs, producing separate (duplicate) device records with the same device name. Therefore, the following topics provide another way to get a *truly* unique name for each device reported in inventory; and recommended best practice is to always leave the **Set Computer Name** check box clear. At the very least, use this convenient AWS method for naming instances only when you deem the risk of duplicate device records in FlexNet Manager Suite to be low enough to ignore (or you are prepared to manually fix any duplicate inventory device records that do occur).

To ensure a device name for each instance that is unique over time, methods vary across platforms, and are detailed in following topics. In summary:

- The easiest method on Windows is to use `Sysprep` when creating your AMI, leaving the name unset so that a new name is provided on instantiation.
- For Linux, if you already provide for unique `Hostname` values on all your instances, no further action is required; and otherwise, a little extra preparation can ensure that the unique `InstanceID` for each instance is also returned in inventory as the `MachineID`, where it differentiates all returned inventory as coming from devices with distinct names.

Schedule specialization

Scheduling inventory collection by the locally-installed FlexNet inventory agent requires that you take either of two different approaches, based the 'life expectancy' of each instance:

- Short-lived, on-demand instances may not have a long enough life cycle for default FlexNet inventory processes to come to full operation. For these short-lived instances, consider a customized default schedule and configuration file in the AMI that will:
  - Provide an upload location (`ManageSoftRL`) for inventory collection
  - Omit any download location, since the instance life-cycle is projected to be too short to require any policy update or system-wide schedule update
  - Trigger inventory collection on start-up, followed immediately by inventory upload to the defined `ManageSoftRL` (this cycle might typically require 3-5 minutes all up, a figure which may change based on other customizations you may want to include in your AMI)
  - Provide a back-up schedule of daily inventory collection, covering off the possibility that, unexpectedly, this instance now needs to run for a longer period.

  Keep in mind, as well, that removing the normal paths for policy update also removes the normal updates to `InventorySettings.xml`, which includes a range of advanced capabilities for FlexNet inventory agent, including Oracle inventory collection and advanced inventory techniques for Microsoft and other vendors. Your strategy therefore includes embedding your most recently downloaded `InventorySettings.xml` in your AMI (and later, updating it as required). For more guidance on customizing short-lived instances, see Configuring an AMI for Short-Lived Instances.
- Instances with longer life spans can be managed exactly as you manage third-party installations on other inventory devices (particularly virtual machines) within your enterprise. These instances do not require a custom schedule, since they can share the same schedule for local inventory collection that applies throughout your enterprise. You

do still need to consider customizations built into the AMI for these instances, such as providing a bootstrap inventory beacon with upload and download locations, as well as giving the instance its unique device name. For more details, see Configuring an AMI for Longer-Life Instances.

Planning your inventory beacons

The final major question is the location of your inventory beacon(s) for uploading inventory data from these instances. Depending on the reliability of network communications between your cloud-based instances and your own enterprise network, you may choose either:

- To run (at least) one instance within your AWS cloud as an inventory beacon, ready 24/7 for any of your instances in the cloud to upload their inventory. This is especially valuable for short-lived instances, where the locally-installed FlexNet inventory agent has no opportunity to retry uploads around any network failures; and where fast network speeds may best support your planned short life-cycle for these instances. The cloud-based inventory beacon can then provide a more rugged upload to your enterprise network, since it has longer up-time and built-in mechanisms for retrying any uploads that are interrupted by transient networking issues. This also enables running the AWS EC2 connector without having to create an IAM User.
- Run an inventory beacon within your enterprise network (or even, potentially, in a demilitarized zone outside your firewall), to which all the cloud-based instances have network access, allowing them to upload their inventory files as and when needed.

Choosing between these alternative places for your inventory beacon may require balancing questions of security and performance against cost. Whichever choice you make, a good practice is to use a DNS alias to identify the inventory beacon, so that you can quickly and easily make changes without disrupting the rest of your infrastructure.

Tip: *As for all communication between your enterprise network and the AWS EC2 network, an Amazon Virtual Private Cloud (VPC) is required for either of the above architectures; and if you require a link from your inventory beacon in the AWS cloud to your central application server in your datacenter, consider a hardware VPN connection as well (see the AWS documentation for details).*

Keep in mind that an inventory beacon does not normally initiate ('push') communication to FlexNet inventory agents installed on your AWS instances. (The only exceptions are for the 'remote execution' tasks known as adoption and zero footprint inventory collection, neither of which you expect to use in a cloud environment.) Since all other communications are initiated by the FlexNet inventory agent ('pull' communications, such as policy updates, collection of self-update packages, and the upload of inventory), this means that your implementation only needs to point the FlexNet inventory agent on each instance to its bootstrap inventory beacon (as described in following topics). You do not need to know details like the IP addresses (public or private) for the instances reporting inventory through the FlexNet inventory agent.

Inventory records and exceptions

After a full inventory import and license reconciliation, you can expect to see records created/updated in FlexNet Manager Suite as follows:

- For every instance where the locally-installed FlexNet inventory agent has uploaded inventory, an inventory device record
- Similarly, for every instance reporting inventory through a third-party tool (such as SCCM), an inventory device record

- For every cloud instance appearing in data from the AWS EC2 connector, regardless of whether it appears in uploaded inventory or not, a cloud instance record in the `CloudServiceInstance` table (and of those, the ones that *also* have an inventory device record show a link to it in the **Cloud Service Provider Inventory** page)
- For dedicated hosts identified through the AWS EC2 connector (and *only* dedicated hosts, not any other kind of host), an inventory device record for the host.

You will *not* find inventory device records for any of the following:

- A cloud instance that does not report inventory
- A host device that is anything other than a dedicate host (such as a default host, or a host for dedicated instances)
- Any instance that you have terminated.

Terminated instances are a special case.

1. Before termination, the running instance may have had an inventory device record (provided that it was reporting inventory).
2. When you terminate the instance, AWS keeps the terminated instance visible through the API for about an hour, and if you have implemented your AWS EC2 connector on an inventory beacon with the default 30 minute schedule, the termination is imported through the AWS EC2 connector, and the record in the `CloudServiceInstance` table is updated with the terminated status. (This record, with its terminated status, is visible in the **Cloud Service Provider Inventory** page, if you set the filters on that page appropriately.)
3. At the next full inventory import (by default, overnight, immediately before the license compliance calculations), *any inventory device record linked to a terminated instance is deleted*, because you do not want terminated instances consuming from your licenses.
4. Of course, the terminated instance no longer reports inventory; but its previously-recorded inventory is still in the inventory database (and in normal processes is not cleaned up for a while). Although the next full import (typically the next night) normally imports records from the inventory database, for terminated instances this is prevented by the terminated flag in the `CloudServiceInstance` table. This prevents the deleted inventory device record from reappearing.

Should you ever need to review the software previously installed on a terminated instance, you can:

1. Navigate to **Discovery & Inventory > Cloud Service Provider Inventory**.
2. Change the default filter (top left) for **Include terminated instances** to `Yes`.
3. Optionally, filter the **Last known state** column to show only `Terminated`.
4. Optionally, use other filters, such as **Instances reported** in an appropriate period, or the **Instance type** and **Instance region** columns, to narrow your search for this instance.
5. Check the **Image ID** column to identify the original image from which this instance was launched (you may need to drag this column out of the column chooser).
6. Through your AWS console, identify this AMI, and inspect its software load and resulting license impacts.

Data integration

Inventory from a hosted instance may come from multiple sources, and must be integrated correctly. Various rules are used depending on the sources being combined:

- AWS EC2 connector + current FlexNet inventory agent are linked by:
  - `CloudServiceProviderID` (displayed as the **Cloud service provider** name in the web interface)
  - **Instance ID**
- AWS EC2 connector + other inventory sources (such as legacy versions of FlexNet inventory agent, or third-party inventory tools, neither of which provides the instance ID) are linked by:
  - **MAC address** (provided that the MAC address is unique in the unmatched inventory and cloud data).

Matching of the incoming data with existing inventory device records (that is, determining whether this is an update to an existing record, or a new record) uses the standard device matching rules, with the **Cloud service provider** and **Instance ID** checked as last priority.

Finally, the merging of data between the AWS EC2 connector and the current FlexNet inventory agent uses these priorities:

1. If one source contains a data point that the other does not, the data is used.
2. If both sources contain the same data point but with different values, the data with the most recent inventory date is used.
3. If different values were imported on the same day, the FlexNet inventory agent is given priority if it has been set as the primary inventory source.
4. Otherwise, the data is used from the most recently created inventory source, with the winning source visible in the inventory device properties as **Last inventory source**.

BYOSL Licensing Considerations

Full software and hardware inventory may be returned from an AWS EC2 instance when either:

- FlexNet inventory agent is locally installed on the instance, usually because it is part of the Amazon Machine Image (AMI) from which the instance was launched
- You have some third-party tool (such as Microsoft SCCM) collecting inventory from your AWS instances, and these results are then being imported into FlexNet Manager Suite.

Whatever the method, when inventory is returned from a running AWS EC2 instance, it has an inventory device record in FlexNet Manager Suite. As always for all inventory devices, the installations found in software inventory for that device consume entitlements from the licenses to which each software record is linked.

However, there are a few special adjustments for inventory from AWS EC2 instances.

Beware of *terminating* short-lived instances

For some special purposes, instances in the AWS EC2 cloud may be launched, run for a short period (from a few minutes to a few hours), and then shut down until needed again.

It is quite possible to configure the AMI from which short-lived instances are launched so that inventory is collected and uploaded through one or more inventory beacons to your FlexNet Manager Suite application server (for details, see Configuring an AMI for Short-Lived Instances).

However, the licensing implications are quite different when you *stop* an EBS-backed instance than when you *terminate*

an instance:

- The assumption is that a *stopped* instance may be restarted when required, and resume operation with (quite likely) the same software installed. It is therefore reasonable to calculate license consumption for such an instance, and, if you use the configuration described in Configuring an AMI for Short-Lived Instances, this happens as usual for virtual machines (whether those are VMs within your enterprise network or instances in the cloud).
- A *terminated* AWS instance cannot be restarted, and if you are choosing BYOSL for instances that you terminate, you should check your license terms carefully and make appropriate provision. The working assumption in FlexNet Manager Suite is that a terminated instance should no longer consume from your software licenses, just as license consumption stops when you uninstall software from a device, or decommission a hardware asset. As described in Collecting Inventory from Instances, this is achieved by permanently deleting any inventory device record that may have previously been created for the instance that is now terminated. (See the same topic for a way of re-discovering the software load for a terminated instance, and hence its potential license implications.)

The complexity of managing licenses on transient instances is why it is more common to purchase these based on a fully-loaded image provided by (and licensed through) AWS. However, if you need repeated runs of a short-lived instance for special purposes, and such an instance must include software for which you provide the license (BYOSL), consider running such an instance as a stop/start instance, rather than terminating it and re-launching it, to simplify your license management.

Thirty-minute inventory and IBM PVU licenses

The default schedule for the AWS EC2 connector is to have it run every 30 minutes. Coincidentally, this happens to be the same interval that IBM requires for hardware checks of devices linked to IBM PVU licenses (when you have IBM's approval to use FlexNet Manager Suite as the source for sub-capacity licensing calculations, for which see the *Sub-Capacity Licensing with IBM PVU* chapter of the *FlexNet Manager Suite System Reference* PDF, available through the title page of online help).

However, each of these coincidental but distinct schedules has its own separate reasons:

- The AWS EC2 connector should run every 30 minutes to allow it to capture transient information that AWS keeps available for only an hour, so that you can track your terminated instances where required. This is the *connector* schedule, and the AWS EC2 connector does not return enough inventory detail to satisfy the IBM requirements.
- The hardware check for IBM PVU licenses must run every 30 minutes as part of your amended contractual agreement with IBM. This is an *inventory* schedule that controls operations of FlexNet inventory agent embedded on an operational instance.

If you are providing IBM PVU licenses for use in the AWS EC2 cloud, *and* have the IBM amended agreement to allow sub-capacity calculations within FlexNet Manager Suite, these two default schedules conveniently satisfy both requirements within AWS. However, be aware of the following requirements:

- IBM PVU licenses in the BYOSL model are not appropriate for very short-life instances that are then terminated (see previous section about the pitfalls of terminating instances with BOYSL). Start/stop operations can be used when necessary, if the instance is maintained and not terminated.
- Persistent (unchanging) inventory device names and matching instance IDs are critical to this license model, allowing proper calculations of peak consumption values over time. For details about setting a unique but persistent device name for inventory, see Configuring an AMI for Longer-Life Instances.
- As always, full hardware and software inventory (from the instance) is required for IBM PVU licensing. Compliance

with your IBM license requires regular software inventory (daily is recommended best practice) as well as the 30-minute hardware check. Use of the FlexNet inventory agent is also mandated by your adjusted license agreement. The techniques in Configuring an AMI for Longer-Life Instances prepare an AMI from which you can launch instances with the operational FlexNet inventory agent embedded, satisfying those requirements.

- As always, the first full software and hardware inventory must be uploaded and processed on the central application server *before* the 30-minute hardware checks take effect. (This allows the calculation of updated rule targets that identify all devices affected by IBM PVU licensing processes.) Typically the relevant compliance calculations take place overnight, after which the revised device policy must be distributed through the inventory beacon hierarchy to affected installations of FlexNet inventory agent, bringing with it the special schedule for the hardware check. All of this process means that in the normal course of events, IBM PVU operations typically are running effectively from the day *after* an instance is launched with PVU-licensed software in operation.

Uncapped calculations except on dedicated hosts

Some license types allow that, where the total license liabilities for many VMs on the same host theoretically exceed factors like the total number of cores available on the host, the summing process is 'capped' or given an upper limit by the actual capacity of the host. For example, if five VMs on a host are each allocated 2 cores, for a total of 10 allocated cores when the host actually only has 8 cores available, then, just as in reality the time-sharing between the VMs limits the cores in use at any moment, so also the license rules cap (or reduce) the calculated figure (10 cores) to the total capacity of the host (8 cores).

Obviously, capping of consumption calculations requires inventory not only of all the VMs on a host, but also of the host itself. However, in the case of AWS EC2 instances, inventory details are not available for any host types other than 'dedicated hosts', over which you have full control. This means that on all other, non-dedicated hosts in AWS EC2, capping of license calculations cannot occur.

All calculations for BOYSL licensing of instances on non-dedicated AWS hosts are based entirely on the inventory taken from the instances themselves, without capping. This does not expose you to any risk of under-licensing (provided that all instances are returning inventory); the only risk is that you may *perhaps* be over-calculating the consumption that *might* apply on a known host with capping applied.

However, this may not be an issue at all. Carefully check your BYOSL terms and use rights. You may find that software publishers already disallow capping in cloud environments, because of the impossibility of tracking instances that may pop up now on this host and now on another, depending on resource allocations within AWS.

3

Images for Different Kinds of Instances

The easiest way to collect inventory from all your cloud-based instances is to make sure that they are launched from Amazon Machine Images (AMI) that already contain everything needed for the collection of FlexNet inventory.

However, the required customization varies somewhat, based on your plans for the resulting instances:

- Instances with short lifespans (from a few minutes to a few hours) need special provision so that they collect and upload inventory as soon as they are launched. For details of ways to achieve this, see Configuring an AMI for Short-Lived Instances.
- Instances with longer lifespans (two days or more, whether or not they are actually running for all of that period) can use a more conventional customization that brings them into the normal policy-based management processes used everywhere by FlexNet Manager Suite. For these, see Configuring an AMI for Longer-Life Instances.

Configuring an AMI for Short-Lived Instances

The requirements for short-lived on demand instances in your AWS EC2 service are somewhat specialized because of the need to capture and upload inventory within a short space of time. This means you need to prepare a separate Amazon Machine Image (AMI) from which to instantiate these short-lived instances. The requirements include:

- Installing the latest approved FlexNet inventory agent into the image
- Adding a customized configuration file that identifies the upload location for inventory
- Installing a customized schedule to manage FlexNet inventory agent in the resulting instances, triggering inventory gathering on start-up (and best practice is to include a backup schedule for further inventory collection in case an instance from this AMI runs for an unexpectedly long time)
- Including the current version of `InventorySettings.xml` that provides advanced functionality for FlexNet inventory agent, since the short lifetime does not allow for normal policy-based downloads
- Ensuring a unique name for each instance created from this AMI.

At this summary level, the requirements are the same for Windows and Linux platforms. In the details, of course, there are platform specifics.

To prepare an AMI for short-lived instances:

1. In the web interface for FlexNet Manager Suite, navigate to **Discovery & Inventory > Inventory Settings**.

   The **Inventory Settings** page displays.

2. Expand the **Inventory agent for download** section.

3. Collect the template configuration file:

   - For a Windows-based AMI, click **Download bootstrapping template file**, and save `mgssetup.ini` to a convenient working folder (such as `C:\temp`). (Do not change the file name.)

   - For a Linux-based AMI, in the *Gathering FlexNet Inventory* PDF (available through the title page of online help), copy the text from *Agent third-party deployment: Sample UNIX Bootstrap Configuration File* and save it as `mgsft_rollout_response` in a convenient working folder (such as `C:\temp`).

4. From the **Inventory agent** drop-down list, select the version of FlexNet inventory agent you want to install in your AMI, and save it to your working folder.

   In general, install the latest available version, subject to your corporate policies. This provides access to the latest functionality. For example, to include advanced inventory for AWS EC2, you must use FlexNet inventory agent 13.2.0 or later (see Appendix 3: Enhanced Inventory Gathered by Agent).

5. In your preferred flat text editor, customize your bootstrapping template file to be used for FlexNet inventory agent in your AWS EC2 environment as follows, saving the edited version in a separate subfolder.

   For Windows, the only *mandatory* change is to identify the upload location for gathered inventory, as described below. (A download location for policy updates is not needed for short-life instances on Windows.) On Linux, FlexNet inventory agent requires both an upload location and a download location. As always, on either platform, experts may also customize other preferences needed for your implementation, as described in the platform-specific topics in the *Gathering FlexNet Inventory* PDF:

   - *Agent third-party deployment: Edit the Configuration File for Microsoft Windows*

   - *Agent third-party deployment: Configure the Bootstrap File for UNIX*.

   The upload (and download) locations require a URL to the host inventory beacon. Best practice is to provide a DNS alias for your chosen inventory beacon, so that when circumstances change, a simple switch of the alias leaves any running and future instances fully operational, without requiring changes to the AMI.

   - For **Windows**, in `mgssetup.ini`:

     a. Locate this section for `Common` preferences, and uncomment (remove the leading semi-colon) and edit the following settings:

```
;=========================================================================
; Registry settings to be created under
; HKLM\Software\ManageSoft Corp\ManageSoft\Common
[Common]
desc0 = UploadSettings\Bootstrap Server\Protocol
val0  = http
```

```
desc1 = UploadSettings\Bootstrap Server\Priority
val1  = 100
desc2 = UploadSettings\Bootstrap Server\AutoPriority
val2  = False
desc3 = UploadSettings\Bootstrap Server\Host
val3  = beacon.fnms.com
desc4 = UploadSettings\Bootstrap Server\Port
val4  = 80
desc5 = UploadSettings\Bootstrap Server\Directory
val5  = /ManageSoftRL/
```

b. Optionally, modify the `Protocol` value if you want to use HTTPS (`val0 = https`).

c. You *must* customize the `Host` preference (shown above as `val3 = beacon.fnms.com`) to your chosen DNS alias (or preferred host setting) and your own enterprise name if the inventory beacon is located in your enterprise network rather than on an AWS EC2 instance. For an inventory beacon hosted on an AWS instance, you can identify its public and private DNS hostnames on the AWS console (select **Instances** in the navigation pane, choose the instance hosting the inventory beacon, and read both its public and private DNS hostnames in the details pane).

Tip: *If you have an inventory beacon on an AWS instance, your installations of FlexNet inventory agent on other instances can be configured to use the private DNS hostname for accessing the inventory beacon (provided that both are within the scope of your VPC). The private hostname has greater stability, particular if the inventory beacon host is stopped and started from time to time.*

d. Optionally, customize the port setting (for example, if you are switching to the HTTPS protocol, the default port is `443`).

Important: *The `Directory` preference is mandatory, and must be set as shown above.*

e. Create a subfolder (such as `ShortLifeAMI`) and save your edited `mgssetup.ini` there. Do not change the file name, which is mandatory.

- For **Linux**, in `mgsft_rollout_response`:

a. In the first section of the file, customize the setting for the download location (see also the comments for the upload location, next):

```
# The initial download location(s) for the installation.
# For example, http://myhost.mydomain.com/ManageSoftDL/
# Refer to the documentation for further details.
MGSFT_BOOTSTRAP_DOWNLOAD=http://beacon.fnms.com:8080/ManageSoftDL/
```

b. In the next section of the file, customize the setting for the reporting (upload) location:

```
# The initial reporting location(s) for the installation.
# For example, http://myhost.mydomain.com/ManageSoftRL/
# Refer to the documentation for further details.
MGSFT_BOOTSTRAP_UPLOAD=http://beacon.fnms.com:8080/ManageSoftRL/
```

You may customize the protocol to HTTPS if required, and omit or customize the port number as required. You *must* customize the host URL (shown above as `beacon.fnms.com`) to your chosen DNS alias (or preferred host setting) and your own enterprise name if the inventory beacon is located in your enterprise network rather than on an AWS EC2 instance. For an inventory beacon hosted on an AWS instance, you can identify its DNS hostname on the AWS console (select **Instances** in the navigation pane, choose the instance hosting the inventory beacon, and read its DNS hostname in the details pane).

Important: *In each case, the trailing `/ManageSoftDL/` (download location) or `/ManageSoftRL/` (upload or reporting location) is mandatory, and must be specified as shown above.*

c. In the last line of the file, switch the value to prevent policy running after installation, as short-lived instances do not have time to wait for policy and its outcomes:

```
MGSFT_RUNPOLICY=0
```

d. Create a subfolder (such as `ShortLifeAMI`) and save your edited `mgsft_rollout_response` there. Do not change the file name, which is mandatory.

e. Unless you already have a strategy and practice that gives every Linux instance a unique name, also create a new text file for your Linux AMI, called `tmpconfig.ini`, with exactly the following two lines:

```
[ManageSoft\Common]
MachineID=
```

This file is deliberately incomplete at this stage, and is to be completed at instance start-up, as described later.

6. For either platform, copy the following and paste it into a plain ASCII text file, and save it in your `ShortLifeAMI` folder as `DefaultSchedule.nds` (this file name is mandatory). Of course, you may choose to unwrap those lines wrapped for readability here.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Schedule PUBLIC "Flexera Inventory Manager schedule"
     "http://www.managesoft.com/dtd/schedule.dtd">

<Schedule NAME="Default Schedule" TIMEDATESTAMP="20180903T044104Z"
SCHEDSCHEMA="60">
        <Event NETWORK="False"
                NAME="Generate Inventory"
                CATCHUP="Never" ID="{F065A477-B500-4611-8340-3BF313FB785B}">
            <LogicalCommand PARAM=" -o UserInteractionLevel=Quiet"
                ACTION="Report" COMPONENT="Tracker"/>
           <Trigger TYPE_PARAM="0" TIMESTART="143950"
                DATESTART="20180903" TYPE="Logon" MAXDELAY="0"/>
        </Event>
      <!-- Following two events are optional (see notes) --!>
        <Event NETWORK="False"
                NAME="Generate Inventory" IDLEDURATION="86400"
                CATCHUP="Never" ID="{F065A477-B500-4611-8340-3BF313FB785B}">
            <LogicalCommand PARAM=" -o UserInteractionLevel=Quiet"
```

```
                ACTION="Report" COMPONENT="Tracker"/>
            <Trigger TYPE_PARAM="1" TIMESTART="001000"
                DATESTART="20180903" TYPE="Daily" REPEAT="21600"
                TIMEWINDOW="3600" DURATION="86400"/>
        </Event>
       <Event NETWORK="False"
                NAME="Upload Client Files"
                CATCHUP="Never" ID="{9CB87BAE-2829-498C-8643-10A9BD3BFA56}">
            <LogicalCommand PARAM="-a"
                ACTION="Upload" COMPONENT="Uploader"/>
            <Trigger TYPE_PARAM="1" TIMESTART="002000"
                DATESTART="20180903" TYPE="Daily" REPEAT="600"
                DURATION="86400"/>
        </Event>
</Schedule>
```

The first event in this schedule triggers machine inventory collection on startup (`TYPE="Logon"`), and the `tracker` component automatically attempts an upload to the bootstrap inventory beacon as soon as inventory gathering is complete. If you are certain that all instances from this image are short-lived, only this event is required. If you are using an EBS-backed instance, and you choose to stop and restart your instance (rather than to terminate and re-launch it), this inventory is checked on each restart.

In addition, two further schedule events provide easy management of cases where an instance proposed to have a short life in fact keeps on running for some longer time.

The second event also generates inventory, and is triggered every 6 hours (`REPEAT="21600"`, with all times expressed in seconds), to start within an hour thereafter (`TIMEWINDOW="3600"`). However, this inventory collection is different than the first case, because here the FlexNet inventory agent does not repeat the inventory collection if it has been successful within the last 24 hours (`IDLEDURATION="86400"`). This means that, effectively, the FlexNet inventory agent checks four times a day whether there has been a successful inventory collection in the last 24 hours, and if not, it runs a new inventory collection.

In a perfect world, the third event is redundant because the `tracker` component uploads inventory as soon as it is collected. This third, separate upload event is a catch-up to work around transient network issues. It checks every 10 minutes to see whether any inventory files have failed to upload (and are therefore still waiting in the staging folder). In normal cases, nothing is waiting, and the `uploader` shuts down immediately, causing negligible load on the instance. But where there have been networking issues, upload is attempted again every 10 minutes until all files are successfully uploaded and removed from the staging folder.

7. On a convenient inventory beacon, navigate to `%CommonAppData%\Flexera Software\Staging\Common\ClientConfiguration`, take a copy of `InventorySettings.xml`, and save it to your `ShortLifeAMI` working folder.

   This completes the preparation of materials for your AMI for instances of this expected life. Now continue to use these materials to prepare the AMI. For both platforms, the high-level process is to customize a running instance, and then save it as an AMI.

   For a Windows-based AMI, the recommended best practice is to run `Sysprep`, as this can easily ensure that each instantiation from this AMI receives a unique device name. For `Sysprep`, AWS offers two different sets of scripts to assist, with separate documentation:

- For a Windows 2016 AMI, use EC2Launch, and see https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/ec2launch.html#ec2launch-sysprep
- For an AMI for an earlier version of Microsoft Windows, use the EC2Config service, and see https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/ami-create-standard.html

For a Linux AMI, the following notes relate to an EBS-backed instance, since this is the simpler process, and the flexibility of the resulting instances is greater (for example, supporting stopping an instance without necessarily terminating it). If you have instead chosen to use an instance store-backed instance, follow the AWS documentation closely, and give it priority where there may be any disparities with this document. Choose from either:

- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/creating-an-ami-ebs.html (preferred, for your EBS-backed instance)
- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/creating-an-ami-instance-store.html (for an instance backed by the instance store).

8. Following the AWS documentation, select (or create), launch and connect to your chosen instance from which you are creating your AMI.
9. To customize your instance:
   a. Ensure that the destination folder exists for your installation of FlexNet inventory agent, and if not create it.

      The default paths for each platform are:

      - On Windows, `C:\Program Files (x86)\Flexera Software\Agent`
      - On Linux, `/opt/managesoft/bin`.

   b. From your local `ShortLifeAMI` working folder, copy these files into the destination folder on your instance:

      - `DefaultSchedule.nds`
      - `InventorySettings.xml`
      - Only on Windows, `mgssetup.ini`
      - Only on Linux, `tmpconfig.ini`.

   c. For Linux only, also copy your amended `mgsft_rollout_response` file to `/var/tmp/` on your instance.
   d. Collect the downloaded installer for FlexNet inventory agent from your working folder, and run the installer on your instance, installing FlexNet inventory agent into the prepared destination folder.

      The installer automatically configures the custom settings declared in the configuration file (either `mgssetup.ini` or `mgsft_rollout_response`, depending on platform). For more information about running the installation, see the *Agent third-party deployment: Details* chapter of the *Gathering FlexNet Inventory* PDF, focusing on the installation topic for your preferred platform.

   e. When installation is complete, in the same destination folder and for both platforms, run the following command to install your default schedule:

```
start ndschedag.exe -t machine .\DefaultSchedule.nds
```

Tip: *When creating your AMI in the following steps, make the name for this image meaningful, such as "Short-life instances", to remind you of the special conditions embodied in the AMI.*

10. **Only for Windows:** complete the remaining steps in the AWS document to `Sysprep` your instance, finishing by clicking **Shutdown with Sysprep** (or, for pre-2016 versions of Windows, click **OK** and confirm **Yes** to run `Sysprep` and shut down the instance).

Important: *If you are using EC2Launch on Windows 2016, in the **Ec2 Launch Settings** dialog, be sure to leave the **Set Computer Name** check box clear. This allows `Sysprep` to clear the computer name from the instance as it shuts down, so that new instances launched from this image each receive a unique device name.*

11. **Only for Linux** (for an EBS-backed image):

    a. Navigate to your instance configuration.

    For example, if you are creating a new instance, select the third tab across the top of your console, showing **3. Configure Instance**.

    b. If necessary, scroll down, and expand the **Advanced Details** panel.

    c. In the **User data** field, add the following two lines **As text** to ensure that each Linux instance reports a unique device name in its inventory (lines are wrapped here for publication, but each command should be entered on a single line):

    ```
    sudo sed -i
      "s/MachineID=/MachineID=$(curl http://169.254.169.254/latest/meta-data/
    instance-id/)/g"
      tmpconfig.ini
    /opt/managesoft/bin/mgsconfig -i /var/tmp/tmpconfig.ini
    ```

    The first line replaces the unfinished `MachineID` setting in the `tmpconfig.ini` file with a setting completed with the Instance ID, which is recovered in the standard way from the AWS metadata for the current image. The second line runs a small FlexNet utility that merges the new value into the standard `config.ini` for the Linux-based FlexNet inventory agent. These two lines of code are run as the instance is being instantiated, before the FlexNet inventory agent is run. The result is that, whenever an instance is launched from the image you are creating, FlexNet inventory agent takes the `MachineID` setting from the `config.ini` file, and reports that unique device name as part of its uploaded inventory.

    d. If your instance is running, stop it now. (Making an AMI from a stopped instance is preferred for lower risk and better stability in your image.)

12. Monitor the **Instances** page in the Amazon EC2 console, until the state for your chosen instance displays `stopped`.

13. Follow your preferred process to create an AMI from that stopped instance.

    For example, on Windows, since you have already installed AWS Tools for Windows PowerShell, you could use the cmdlet `New-EC2Image`. For Linux, in the navigation pane, you can choose **Instances**, select your instance, and then choose **Actions > Image > Create Image**, and provide the necessary details. For more information on creating AMIs, see either of:

- https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/Creating_EBSbacked_WinAMI.html
- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/creating-an-ami-ebs.html.

This AMI is now ready to instantiate short-lived instances. As soon as each instance is spawned, the embedded FlexNet inventory agent gathers inventory from the instance, and immediately uploads it to the bootstrap inventory beacon. Depending on the resources on each instance, this process may take around 2-5 minutes, after which the instance can be stopped (but remember, not terminated if you wish to track related software licenses in FlexNet Manager Suite).

Configuring an AMI for Longer-Life Instances

An AWS EC2 instance that runs for a longer period (such as at least a day or two) can take part in the normal processes for inventory collection that you use for any virtual machines, whether on-premises or in the cloud. The requirements include:

- Installing the latest approved FlexNet inventory agent into the image
- Adding a customized configuration file that identifies both the download location for requesting the initial policy for FlexNet inventory agent on the instance, and the upload location for inventory
- Ensuring a unique name for each instance created from this AMI.

Both the operational schedule for FlexNet inventory agent, and the latest `InventorySettings.xml` providing enhanced inventory-gathering functionality, are delivered automatically as part of policy. However, this can take a couple of days in the worst case; so if you wish you may also include a start-up schedule and `InventorySettings.xml` in the image to shorten the time to full functionality of FlexNet inventory agent. If you do so, both of these are over-written automatically once policy is fully operational. (You can find some notes about these additional documents in Configuring an AMI for Short-Lived Instances.)

These guidelines cover both Windows and Linux instances, so skip the parts that do not apply to your platform.

To prepare an AMI for longer-life instances:

1. In the web interface for FlexNet Manager Suite, navigate to **Discovery & Inventory > Inventory Settings**.

   The **Inventory Settings** page displays.

2. Collect the template configuration file:

   - For a Windows-based AMI, click **Download bootstrapping template file**, and save `mgssetup.ini` to a convenient working folder (such as `C:\temp`). (Do not change the file name.)
   - For a Linux-based AMI, in the *Gathering FlexNet Inventory* PDF (available through the title page of online help), copy the text from *Agent third-party deployment: Sample UNIX Bootstrap Configuration File* and save it as `mgsft_rollout_response` in a convenient working folder (such as `C:\temp`).

3. From the **Inventory agent** drop-down list, select the version of FlexNet inventory agent you want to install in your AMI, and save it to your working folder.

   In general, install the latest available version, subject to your corporate policies. This provides access to the latest

functionality. For example, to include advanced inventory for AWS EC2, you must use FlexNet inventory agent 13.2.0 or later (see Appendix 3: Enhanced Inventory Gathered by Agent).

4. In your preferred flat text editor, customize your bootstrapping template file to be used for FlexNet inventory agent in your AWS EC2 environment as follows, saving the edited version in a separate subfolder.

   For Windows, the only *mandatory* change is to identify the download location for requesting policy, as described below. (On Windows, all other requirements are delivered through policy; but for safety we'll also provide an upload location.) On Linux, FlexNet inventory agent *requires* both an upload location and a download location. As always, on either platform, experts may also customize other preferences needed for your implementation, as described in the platform-specific topics in the *Gathering FlexNet Inventory* PDF:

   - *Agent third-party deployment: Edit the Configuration File for Microsoft Windows*
   - *Agent third-party deployment: Configure the Bootstrap File for UNIX*.

   The download (and upload) locations require a URL to the host inventory beacon. Best practice is to provide a DNS alias for your chosen inventory beacon, so that when circumstances change, a simple switch of the alias leaves any running and future instances fully operational, without requiring changes to the AMI.

   - For **Windows**, in `mgssetup.ini`:

     a. Locate this section for `Common` preferences, and uncomment (remove the leading semi-colon) and edit the following settings:

```
;=========================================================================
; Registry settings to be created under
; HKLM\Software\ManageSoft Corp\ManageSoft\Common
[Common]
desc0 = UploadSettings\Bootstrap Server\Protocol
val0  = http
desc1 = UploadSettings\Bootstrap Server\Priority
val1  = 100
desc2 = UploadSettings\Bootstrap Server\AutoPriority
val2  = False
desc3 = UploadSettings\Bootstrap Server\Host
val3  = beacon.fnms.com
desc4 = UploadSettings\Bootstrap Server\Port
val4  = 80
desc5 = UploadSettings\Bootstrap Server\Directory
val5  = /ManageSoftRL/


desc6 = DownloadSettings\Bootstrap Server\Protocol
val6  = http
desc7 = DownloadSettings\Bootstrap Server\Priority
val7  = 100
desc8 = DownloadSettings\Bootstrap Server\AutoPriority
val8  = False
desc9 = DownloadSettings\Bootstrap Server\Host
val9  = beacon.fnms.com
desc10 = DownloadSettings\Bootstrap Server\Port
```

```
val10  = 80
desc11 = DownloadSettings\Bootstrap Server\Directory
val11  = /ManageSoftDL/
```

b. Optionally, modify the `Protocol` values if you want to use HTTPS (`val0 = https`, and `val6 = https`).

c. You *must* customize the `Host` preference (shown above as `val3 = beacon.fnms.com` and `val9 = beacon.fnms.com`) to your chosen DNS alias (or preferred host setting) and your own enterprise name if the inventory beacon is located in your enterprise network rather than on an AWS EC2 instance. For an inventory beacon hosted on an AWS instance, you can identify its DNS hostname on the AWS console (select **Instances** in the navigation pane, choose the instance hosting the inventory beacon, and read its DNS hostname in the details pane).

Tip: *If you have an inventory beacon on an AWS instance, your installations of FlexNet inventory agent on other instances can be configured to use the private DNS hostname for accessing the inventory beacon (provided that both are within the scope of your VPC). The private hostname has greater stability, particular if the inventory beacon host is stopped and started from time to time.*

d. Optionally, customize the port settings at `val4` and `val10` (for example, if you are switching to the HTTPS protocol, the default port is `443`).

Important: *The `Directory` preference is mandatory, and in each case must be set as shown above.*

e. Create a subfolder (such as `LongLifeAMI`) and save your edited `mgssetup.ini` there. Do not change the file name, which is mandatory.

- For **Linux**, in `mgsft_rollout_response`:

a. In the first section of the file, customize the setting for the download location (see also the comments for the upload location, next):

```
# The initial download location(s) for the installation.
# For example, http://myhost.mydomain.com/ManageSoftDL/
# Refer to the documentation for further details.
MGSFT_BOOTSTRAP_DOWNLOAD=http://beacon.fnms.com:8080/ManageSoftDL/
```

b. In the next section of the file, customize the setting for the reporting (upload) location:

```
# The initial reporting location(s) for the installation.
# For example, http://myhost.mydomain.com/ManageSoftRL/
# Refer to the documentation for further details.
MGSFT_BOOTSTRAP_UPLOAD=http://beacon.fnms.com:8080/ManageSoftRL/
```

You may customize the protocol to HTTPS if required, and omit or customize the port number as required. You *must* customize the host URL (shown above as `beacon.fnms.com`) to your chosen DNS alias (or preferred host setting) and your own enterprise name if the inventory beacon is located in your enterprise network rather than on an AWS EC2 instance. For an inventory beacon hosted on an AWS instance, you can identify its DNS hostname on the AWS console (select **Instances** in the navigation pane, choose the instance hosting the inventory beacon, and read its DNS hostname in the details pane).

Important: *In each case, the trailing `/ManageSoftDL/` (download location) or `/ManageSoftRL/` (upload or reporting location) is mandatory, and must be specified as shown above.*

c. Create a subfolder (such as `LongLifeAMI`) and save your edited `mgsft_rollout_response` there. Do not change the file name, which is mandatory.

d. Unless you already have a strategy and practice that gives every Linux instance a unique name, also create a new text file for your Linux AMI, called `tmpconfig.ini`, with exactly the following two lines:

```
[ManageSoft\Common]
MachineID=
```

This file is deliberately incomplete at this stage, and is to be completed at instance start-up, as described later.

This completes the preparation of materials for your AMI for instances of this expected life. Now continue to use these materials to prepare the AMI. For both platforms, the high-level process is to customize a running instance, and then save it as an AMI.

For a Windows-based AMI, the recommended best practice is to run `Sysprep`, as this can easily ensure that each instantiation from this AMI receives a unique device name. For `Sysprep`, AWS offers two different sets of scripts to assist, with separate documentation:

- For a Windows 2016 AMI, use EC2Launch, and see https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/ec2launch.html#ec2launch-sysprep
- For an AMI for an earlier version of Microsoft Windows, use the EC2Config service, and see https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/ami-create-standard.html

For a Linux AMI, the following notes relate to an EBS-backed instance, since this is the simpler process, and the flexibility of the resulting instances is greater (for example, supporting stopping an instance without necessarily terminating it). If you have instead chosen to use an instance store-backed instance, follow the AWS documentation closely, and give it priority where there may be any disparities with this document. Choose from either:

- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/creating-an-ami-ebs.html (preferred, for your EBS-backed instance)
- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/creating-an-ami-instance-store.html (for an instance backed by the instance store).

5. Following the AWS documentation, select (or create), launch and connect to your chosen instance from which you are creating your AMI.

6. To customize your instance:

a. Ensure that the destination folder exists for your installation of FlexNet inventory agent, and if not create it.

The default paths for each platform are:

- On Windows, `C:\Program Files (x86)\Flexera Software\Agent`
- On Linux, `/opt/managesoft/bin`.

b. From your local `ShortLifeAMI` working folder, copy these files as shown onto your instance:

- For Windows, copy `mgssetup.ini` into the destination folder `C:\Program Files (x86)\Flexera Software\Agent`
- For Linux, copy `mgsft_rollout_response` into `/var/tmp/` on your instance.

c. Collect the downloaded installer for FlexNet inventory agent from your working folder, and run the installer on your instance, installing FlexNet inventory agent into the prepared destination folder.

The installer automatically configures the custom settings declared in the configuration file (either `mgssetup.ini` or `mgsft_rollout_response`, depending on platform). For more information about running the installation, see the *Agent third-party deployment: Details* chapter of the *Gathering FlexNet Inventory* PDF, focusing on the installation topic for your preferred platform.

Tip: *When creating your AMI in the following steps, make the name for this image meaningful, such as "Long-life instances", to remind you of the special conditions embodied in the AMI.*

7. **Only for Windows:** complete the remaining steps in the AWS document to `Sysprep` your instance, finishing by clicking **Shutdown with Sysprep** (or, for pre-2016 versions of Windows, click **OK** and confirm **Yes** to run `Sysprep` and shut down the instance).

Important: *If you are using EC2Launch on Windows 2016, in the **Ec2 Launch Settings** dialog, be sure to leave the **Set Computer Name** check box clear. This allows `Sysprep` to clear the computer name from the instance as it shuts down, so that new instances launched from this image each receive a unique device name.*

8. **Only for Linux** (for an EBS-backed image):

a. Navigate to your instance configuration.

For example, if you are creating a new instance, select the third tab across the top of your console, showing **3. Configure Instance**.

b. If necessary, scroll down, and expand the **Advanced Details** panel.

c. In the **User data** field, add the following two lines **As text** to ensure that each Linux instance reports a unique device name in its inventory (lines are wrapped here for publication, but each command should be entered on a single line):

```
sudo sed -i
  "s/MachineID=/MachineID=$(curl http://169.254.169.254/latest/meta-data/
instance-id/)/g"
  tmpconfig.ini
/opt/managesoft/bin/mgsconfig -i /var/tmp/tmpconfig.ini
```

The first line replaces the unfinished `MachineID` setting in the `tmpconfig.ini` file with a setting completed with the Instance ID, which is recovered in the standard way from the AWS metadata for the current image. The second line runs a small FlexNet utility that merges the new value into the standard `config.ini` for the Linux-based FlexNet inventory agent. These two lines of code are run as the instance is being instantiated, before the FlexNet inventory agent is run. The result is that, whenever an instance is launched from the image you are creating, FlexNet inventory agent takes the `MachineID` setting from the `config.ini` file, and reports that unique device name as part of its uploaded inventory.

   d. If your instance is running, stop it now. (Making an AMI from a stopped instance is preferred for lower risk and better stability in your image.)

9. Monitor the **Instances** page in the Amazon EC2 console, until the state for your chosen instance displays `stopped`.

10. Follow your preferred process to create an AMI from that stopped instance.

For example, on Windows, since you have already installed AWS Tools for Windows PowerShell, you could use the cmdlet `New-EC2Image`. For Linux, in the navigation pane, you can choose **Instances**, select your instance, and then choose **Actions > Image > Create Image**, and provide the necessary details. For more information on creating AMIs, see either of:

- https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/Creating_EBSbacked_WinAMI.html
- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/creating-an-ami-ebs.html.

This AMI is now ready to instantiate longer-life instances. As soon as each instance is spawned, FlexNet inventory agent asks its bootstrap inventory beacon for policy. Once downloaded, its device policy triggers delivery of your global schedule for inventory collection by the locally-installed FlexNet inventory agent, as well as the latest `InventorySettings.xml` file that brings enhanced functionality.

4

Appendices: Technical Data

The following topics do not include content you need for set-up or operation of the AWS EC2 connector. These are the deep-down technical data that you can ignore until you have a real need.

Appendix 1: Cmdlets in the AWS EC2 connector

The connection from FlexNet Manager Suite to the AWS EC2 service relies on the AWS Tools for Windows PowerShell (as described in Prerequisites and Setting Up). Within that toolset, the AWS EC2 connector uses the following PowerShell cmdlets:

- `Get-AWSRegion` — To identify the available AWS geographic regions where your instances may be running
- `Get-EC2Instance` — To return the list of the instances (virtual machines) you own, and then collect summary inventory details on each one
- `Get-EC2Host` — To identify any dedicated hosts you have available in each region
- `Get-EC2ReservedInstance` — To return inventory details on any instance/capacity you have reserved in an Availability Zone
- `Get-IAMUser` — Used as a test for connection success
- `Get-IAMPolicy` — To retrieve information about the specified managed policy, including the policy's default version and the total number of IAM users, groups, and roles to which the policy is attached
- `Get-IAMPolicyVersion` — To retrieve information about the specified version of the specified managed policy, including the policy document
- `Get-IAMAttachedRolePolicyList` — Lists all managed policies that are attached to the specified IAM role
- `Get-IAMAttachedUserPolicyList` — Lists all managed policies that are attached to the specified IAM user
- `Get-IAMGroupForUser` — Lists the IAM groups that the specified IAM user belongs to
- `Get-STSCallerIdentity` — To return details about the IAM user or role whose credentials are used to call the

operation

- `Get-IAMRole` — To retrieves information about the specified role and the role's trust policy that grants permission to assume the role
- `Use-STSRole` — To return a set of temporary security credentials that you can use to access AWS resources that you might not normally have access to
- `Set-AWSProxy` — Specifies the proxy setting for the following interactions through other cmdlets

Appendix 2: Data Imported by AWS EC2 connector

The AWS EC2 connector returns information about each AWS instance, and its host. All instances are subsequently displayed in the **Cloud Service Provider Inventory** page. Hosts are never displayed in this page, as in general hosting is variable and controlled by AWS.

Tip: *One exception where you have direct control is the 'dedicated host'. Like all other host types, dedicated hosts do not appear in the* ***Cloud Service Provider Inventory*** *listing; but they do automatically appear in inventory device listings, such as the* ***All Inventory*** *page, where you can search for them using the* ***Hosted in*** *property. Any instances from the* ***Cloud Service Provider Inventory*** *listing that are hosted on your dedicated host also display a link to the dedicated host properties in the* ***Host*** *column.*

Instance data

Tip: *All kinds of instances return this set of data, whether they are virtual machines or bare metal instances. The latter return an* `Instance type` *starting with* `i3.`*.*

If the data gathered through the AWS EC2 connector matches inventory collected from another source, the **Cloud Service Provider Inventory** page includes a hyperlink to open the inventory device properties page for the instance. Putting that another way, inventory device records are *not* created when the AWS EC2 connector is the sole source of information; but they *are* created when inventory is returned from another source that covers the cloud instance (one possible example is having FlexNet inventory agent running locally on the instance). In this latter case, linked records exist in both the `ComplianceComputer` and `CloudServiceInstance` tables.

Some data is additional to the normal content collected as inventory. For example, the `InstanceTenancyID`, used for identifying whether the current device is a `Dedicated instance`, or is an instance running `On dedicated host`, is imported into the `CloudServiceInstance` table, and, as **Availability type**, is visible in both the **Cloud Service Provider Inventory** page and the **Cloud Hosting** tab of the inventory device properties (where inventory exists for the instance).

Tip: *AWS "bare metal instances" behave like other instances in terms of data gathered and records created.*

When there is an inventory device record for an instance, the inventory-related values imported through the AWS EC2 connector (and shown below in alphabetical order) are displayed in the separate **Cloud Hosting** tab of the inventory device properties:

- `Availability type`
- `Availability Zone`
- `Cores`
- `Creation date` (and time)
- `Host ID` (use to link to host)
- `Image ID` (for the AMI, or Amazon Machine Image, from which the instance was instantiated)
- `Instance ID`
- `Instance region`
- `Instance type`
- `Last known state`
- `Lifecycle mode` (displayed as **Lifecycle** in the **Cloud Service Provider Inventory** page, and as **Purchased option** in the **Cloud Hosting** tab of the inventory device properties)
- `MAC address`
- `Network ID`
- `Threads per core`.

Host data

If the host is a "dedicated host" (which means that your enterprise has exclusive use and detailed control), an inventory device record is automatically created for the dedicated host in the `ComplianceComputer` table (with the resulting property sheet). Separate inventory device records are *not* created for any other kind of host; and hosts (other than dedicated hosts) are *not* listed separately on the **Cloud Service Provider Inventory** page.

Tip: *For dedicated hosts, the following properties are displayed on the* ***Cloud Hosting*** *tab of the inventory device properties.*

The following properties (listed alphabetically) are collected for hosts of all types through the AWS EC2 connector:

- `Availability Zone`
- `Cores`
- `Creation time`
- `Host ID`
- `Instance region`
- `Instance type`
- `SocketsVcpu`.

Appendix 3: Enhanced Inventory Gathered by Agent

As well as the high-level data gathered from the AWS EC2 connection, FlexNet inventory agent (version 13.1 or later) gathers addition inventory data for each AWS instance where FlexNet inventory agent is locally installed. To do so, FlexNet inventory agent accesses two AWS-internal URLs:

- `http://169.254.169.254/latest/meta-data/instance-id` returns the unique identifier for the AWS EC2 instance.
- `http://169.254.169.254/latest/dynamic/instance-identity/document` returns a JSON-formatted value containing data about the instance where FlexNet inventory agent is locally installed. For example:

```
{
    "devpayProductCodes" : null,
    "marketplaceProductCodes" : [ "1abc2defghijklm3nopqrs4tu" ],
    "availabilityZone" : "us-west-2b",
    "privateIp" : "10.158.112.84",
    "version" : "2017-09-30",
    "instanceId" : "i-1234567890abcdef0",
    "billingProducts" : null,
    "instanceType" : "t2.micro",
    "accountId" : "123456789012",
    "imageId" : "ami-5fb8c835",
    "pendingTime" : "2018-11-19T16:32:11Z",
    "architecture" : "x86_64",
    "kernelId" : null,
    "ramdiskId" : null,
    "region" : "us-west-2"
}
```

These two data points are then embedded within the normal `.ndi` inventory file as a pseudo-hardware WMI class called `MGS_AWSComputerSystem`. The class looks similar to this example (lines wrapped for presentation):

```
<Hardware class="MGS_AWSComputerSystem" Name="ec2amaz-vrkiu0p" Evidence="metadata">
    <Property Name="instance-id" Value="i-1234567890abcdef0"
              Evidence="/latest/meta-data/instance-id"/>
    <Property Name="document" Value="ewogICJwcml2YXRlSXAiIDogIjE3Mi4zMS4yLjExNiIs...."
              Evidence="/latest/dynamic/instance-identity/document"/>
</Hardware>
```

After a full inventory import, these details are eventually saved in the `CloudServiceInstance` table in the compliance database.

V

Microsoft Azure Connector

This part introduces the FlexNet connector to Microsoft Azure, enabling you to collect Azure virtual machine inventory data, managed by **Azure Resource Manager**.

The connector to Azure is only a part of a complete "bring your own software license", or BYOSL management strategy for instances (typically, virtual machines that are tracked as inventory devices) that are hosted for you by this cloud service provider. By itself, the connector does not gather sufficient data to allow for license management: for example, it does not gather software inventory for the applications that are running on your cloud-hosted devices.

For that reason, this part covers more than the connector itself. As well as details of prerequisites, set up, and data gathering with the connector, the part includes some insights into:

- Processes for gathering *complete* hardware and software inventory for your cloud-based instances
- Considerations for license management

In fact, using those guidelines, you *could* configure inventory gathering in Azure without using the connector at all. Provided that you are deploying the latest FlexNet inventory agent (13.2.x or later), inventory alone even returns the cloud service provider. What the connector adds is:

- Automatically setting the **Hosted in** property in the inventory device records created when inventory is returned
- Adding the cloud service provider and instance details when these are missing from inventory collected by legacy versions of FlexNet inventory agent, or from third-party inventory sources
- Populating the **Cloud Service Provider Inventory** page with records of instances currently running in your Azure environment, including a few properties that are additional to the normal hardware and software inventory
- Importing permanent records of instances that were previously running but that have now been terminated (and for which, as a result, any prior inventory devices records have now been deleted)
- Triggering automatic deletion (completed at the next full import and compliance calculation) of any inventory device record linked to an instance that is now terminated.

This part focuses on background about the connector, and additional material you need in order to build license management around the connector. You will not find details about making the actual connection here.

Note: *The Microsoft Azure inventory connector does not collect virtual machine inventory for virtual machines managed by Microsoft's classic deployment model. For these devices we recommend you use an alternative such as the inventory*

spreadsheet upload, the Business Importer or simply through editing manually.

Note: *Containers are not currently supported.*

Step-by-step instructions for configuring the connection to Azure are available in the online help at ***FlexNet Manager Suite Online Help > Inventory Beacons > Inventory Systems Page > Connecting to External Inventory Systems > Managing PowerShell Connections > Managing Azure Connections***.

1

Prerequisites and Setting Up

The connector to Microsoft Azure is configured on a convenient inventory beacon, for which there are the following prerequisites:

- The operating system is either:
  - Windows Server 2008 R2 SP1 or later
  - Windows 7 SP1 or later.
- FlexNet Beacon software has been updated to release 13.2.x.x (2018 R1) or later
- PowerShell 5.1 or later is running
- The Azure adapter requires a service principal identity with read permission to **Microsoft.Compute/virtualMachines**
- FlexNet inventory agent version 13.2.x.x (2019 R1) or later
- The PowerShell execution policy is set to `RemoteSigned`
- A web browser is available
- AzureRM for Windows PowerShell version 5.1 or later is installed

---

Tip: *To check the version installed on your inventory beacon:*

1. *As administrator, run AzureRM for Windows PowerShell.*
2. *Execute the `$psversiontable` cmdlet.*

New versions are available for download from https://www.powershellgallery.com/packages/AzureRM.

- You must log onto the inventory beacon, and run FlexNet Beacon, using an account with administrator privileges.

On the Azure side, you must first create:

- A service principal identity

Step-by-step instructions for configuring the connection to Azure are available in the online help at ***FlexNet Manager Suite Online Help > Inventory Beacons > Inventory Systems Page > Connecting to External Inventory Systems > Managing PowerShell Connections > Managing Azure Connections***.

2

Thinking about Inventory and Licensing

In many ways, collecting inventory and working out license consumption for instances in Microsoft Azure is much the same as licensing virtual machines on host servers sitting within the datacenter in your own offices. However, there are some differences that are worth thinking about. For example, within your own datacenter, you have full control of, and presumably full inventory for, the host server as well as the VMs running on it; and for some license types, host details are relevant even when licensing software on a virtual machine. Depending on what you purchase, this may not be the case in Microsoft Azure.

The following topics bring together some points to consider, first about inventory collection, and then about licensing.

Collecting Inventory from Instances

The connector to Microsoft Azure gathers useful information, but it cannot gather information valuable for license consumption calculations from each of the instances running in your Azure cloud. To gather useful inventory from all instances, best practice is to include the FlexNet inventory agent (release 13.2.`buildNumber` or later) in the Microsoft Azure managed VM image from which your devices are instantiated. For details, see the Azure documentation *Create a managed image of a generalized VM in Azure* for Windows, or for Linux see *How to create an image of a virtual machine or VHD*.

Unique device naming is critical

It is critical to your license management in Microsoft Azure to ensure that each instance is *uniquely* named on instantiation. If you do not do this, incoming inventory reports are not differentiated, and are interpreted as updated inventory from a single device, named with the default device name provided by the base image.

By default, the internal name of a virtual machine is the same as the external name given to it when it is created. While you can change the internal name of a virtual machine, you cannot change the external name of the virtual machine as displayed in the Azure portal.

Resource names for virtual machines are immutable. So, you will need to redeploy your virtual machine. You can do this by deleting the current virtual machine, while maintaining the current virtual machine disks, and then creating a new virtual machine using the correct name and point it to the previously maintained disks.

To ensure a device name for each instance that is unique over time, methods vary across platforms, and are detailed in following topics. In summary:

- The easiest method on Windows is to use `Sysprep` when creating your managed VM image, leaving the name unset so that a new name is provided on instantiation.
- For Linux, if you already provide for unique `Hostname` values on all your instances, no further action is required; and otherwise, a little extra preparation can ensure that the unique `InstanceID` for each instance is also returned in inventory as the `MachineID`, where it differentiates all returned inventory as coming from devices with distinct names.

Schedule specialization

Scheduling inventory collection by the locally-installed FlexNet inventory agent requires:

- Provide an upload location (`ManageSoftRL`) for inventory collection
- Omit any download location, since the instance life-cycle is projected to be too short to require any policy update or system-wide schedule update
- Trigger inventory collection on start-up, followed immediately by inventory upload to the defined `ManageSoftRL` (this cycle might typically require 3-5 minutes all up, a figure which may change based on other customizations you may want to include in your managed VM image)
- Provide a back-up schedule of daily inventory collection.
- Keep in mind, as well, that removing the normal paths for policy update also removes the normal updates to `InventorySettings.xml`, which includes a range of advanced capabilities for FlexNet inventory agent, including Oracle inventory collection and advanced inventory techniques for Microsoft and other vendors. Your strategy therefore includes embedding your most recently downloaded `InventorySettings.xml` in your managed VM image (and later, updating it as required). For more guidance on customizing instances, see Configuring an Azure VM Image for Instances.

Planning your inventory beacons

The final major question is the location of your inventory beacon(s) for uploading inventory data from these instances. Depending on the reliability of network communications between your cloud-based instances and your own enterprise network, you may choose either:

- To run (at least) one instance within your Azure cloud as an inventory beacon, ready 24/7 for any of your instances in the cloud to upload their inventory. This is especially valuable for instances, where the locally-installed FlexNet inventory agent has no opportunity to retry uploads around any network failures; and where fast network speeds may best support your planned life-cycle for these instances. The cloud-based inventory beacon can then provide a more rugged upload to your enterprise network, since it has longer up-time and built-in mechanisms for retrying any uploads that are interrupted by transient networking issues.
- Run an inventory beacon within your enterprise network (or even, potentially, in a demilitarized zone outside your firewall), to which all the cloud-based instances have network access, allowing them to upload their inventory files as and when needed.

Choosing between these alternative places for your inventory beacon may require balancing questions of security and performance against cost. Whichever choice you make, a good practice is to use a DNS alias to identify the inventory

beacon, so that you can quickly and easily make changes without disrupting the rest of your infrastructure.

Tip: *As for all communication between your enterprise network and the Microsoft Azure network, an Amazon Virtual Private Cloud (VPC) is required for either of the above architectures; and if you require a link from your inventory beacon in the Azure cloud to your central application server in your datacenter, consider a hardware VPN connection as well (see the Microsoft Azure documentation for details).*

Keep in mind that an inventory beacon does not normally initiate ('push') communication to FlexNet inventory agents installed on your Microsoft Azure instances. (The only exceptions are for the 'remote execution' tasks known as adoption and zero footprint inventory collection, neither of which you expect to use in a cloud environment.) Since all other communications are initiated by the FlexNet inventory agent ('pull' communications, such as policy updates, collection of self-update packages, and the upload of inventory), this means that your implementation only needs to point the FlexNet inventory agent on each instance to its bootstrap inventory beacon (as described in following topics). You do not need to known details like the IP addresses (public or private) for the instances reporting inventory through the FlexNet inventory agent.

Inventory records and exceptions

After a full inventory import and license reconciliation, you can expect to see records created/updated in FlexNet Manager Suite as follows:

- For every instance where the locally-installed FlexNet inventory agent has uploaded inventory, an inventory device record
- Similarly, for every instance reporting inventory through a third-party tool (such as SCCM), an inventory device record
- For every cloud instance appearing in data from the Azure PowerShell connector, regardless of whether it appears in uploaded inventory or not, a cloud instance record in the `CloudServiceInstance` table (and of those, the ones that *also* have an inventory device record show a link to it in the **Cloud Service Provider Inventory** page)
- For dedicated hosts identified through the Azure PowerShell connector (and *only* dedicated hosts, not any other kind of host), an inventory device record for the host.

You will *not* find inventory device records for any of the following:

- A cloud instance that does not report inventory
- A host device that is anything other than a dedicate host (such as a default host, or a host for dedicated instances)
- Any instance that you have terminated.

Terminated instances are a special case.

1. Before termination, the running instance may have had an inventory device record (provided that it was reporting inventory).
2. When you terminate the instance, Microsoft Azure keeps the terminated instance visible through the API for about an hour, and if you have implemented your Azure PowerShell connector on an inventory beacon with the default 30 minute schedule, the termination is imported through the connector, and the record in the `CloudServiceInstance` table is updated with the terminated status. (This record, with its terminated status, is visible in the **Cloud Service Provider Inventory** page, if you set the filters on that page appropriately.)
3. At the next full inventory import (by default, overnight, immediately before the license compliance calculations),

any inventory device record linked to a terminated instance is deleted, because you do not want terminated instances consuming from your licenses.

4. Of course, the terminated instance no longer reports inventory; but its previously-recorded inventory is still in the inventory database (and in normal processes is not cleaned up for a while). Although the next full import (typically the next night) normally imports records from the inventory database, for terminated instances this is prevented by the terminated flag in the `CloudServiceInstance` table. This prevents the deleted inventory device record from reappearing.

Should you ever need to review the software previously installed on a terminated instance, you can:

1. Navigate to **Discovery & Inventory > Cloud Service Provider Inventory**.
2. Change the default filter (top left) for **Include terminated instances** to `Yes`.
3. Optionally, filter the **Last known state** column to show only `Terminated`.
4. Optionally, use other filters, such as **Instances reported** in an appropriate period, or the **Instance type** and **Instance region** columns, to narrow your search for this instance.
5. Check the **Image ID** column to identify the original image from which this instance was launched (you may need to drag this column out of the column chooser).
6. Through your Azure console, identify this managed VM image, and inspect its software load and resulting license impacts.

Data integration

Inventory from a hosted instance may come from multiple sources, and must be integrated correctly. Various rules are used depending on the sources being combined:

- PowerShell connector + current FlexNet inventory agent are linked by:
  - `CloudServiceProviderID` (displayed as the **Cloud service provider** name in the web interface)

Matching of the incoming data with existing inventory device records (that is, determining whether this is an update to an existing record, or a new record) uses the standard device matching rules, with the **Cloud service provider** and **Instance ID** checked as last priority.

Finally, the merging of data between the PowerShell connector and the current FlexNet inventory agent uses these priorities:

1. If one source contains a data point that the other does not, the data is used.
2. If both sources contain the same data point but with different values, the data with the most recent inventory date is used.
3. If different values were imported on the same day, the FlexNet inventory agent is given priority if it has been set as the primary inventory source.
4. Otherwise, the data is used from the most recently created inventory source, with the winning source visible in the inventory device properties as **Last inventory source**.

BYOSL Licensing Considerations

Full software and hardware inventory may be returned from an Microsoft Azure instance when either:

- FlexNet inventory agent is locally installed on the instance, usually because it is part of the Azure VM Image from which the instance was launched
- You have some third-party tool (such as Microsoft SCCM) collecting inventory from your Azure instances, and these results are then being imported into FlexNet Manager Suite.

Whatever the method, when inventory is returned from a running Microsoft Azure instance, it has an inventory device record in FlexNet Manager Suite. As always for all inventory devices, the installations found in software inventory for that device consume entitlements from the licenses to which each software record is linked.

However, there are a few special adjustments for inventory from Microsoft Azure instances.

Beware of *terminating* instances

For some special purposes, instances in the Microsoft Azure cloud may be launched, run for a short period (from a few minutes to a few hours), and then shut down until needed again.

It is quite possible to configure the Azure VM Image from which instances are launched so that inventory is collected and uploaded through one or more inventory beacons to your FlexNet Manager Suite application server (for details, see Configuring an Azure VM Image for Instances).

However, the licensing implications are quite different when you *stop* an instance than when you *terminate* an instance:

- The assumption is that a *stopped* instance may be restarted when required, and resume operation with (quite likely) the same software installed. It is therefore reasonable to calculate license consumption for such an instance, and, if you use the configuration described in Configuring an Azure VM Image for Instances, this happens as usual for virtual machines (whether those are VMs within your enterprise network or instances in the cloud).
- A *terminated* instance cannot be restarted, and if you are choosing BYOSL for instances that you terminate, you should check your license terms carefully and make appropriate provision. The working assumption in FlexNet Manager Suite is that a terminated instance should no longer consume from your software licenses, just as license consumption stops when you uninstall software from a device, or decommission a hardware asset. As described in Collecting Inventory from Instances, this is achieved by permanently deleting any inventory device record that may have previously been created for the instance that is now terminated. All resources attached to the terminated instance will remain and you can then manually delete these resources as desired. (See the same topic for a way of re-discovering the software load for a terminated instance, and hence its potential license implications.)

The complexity of managing licenses on transient instances is why it is more common to purchase these based on a fully-loaded image provided by (and licensed through) Microsoft Azure. However, if you need repeated runs of an instance for special purposes, and such an instance must include software for which you provide the license (BYOSL), consider running such an instance as a stop/start instance, rather than terminating it and re-launching it, to simplify your license management.

Thirty-minute inventory and IBM PVU licenses

The default schedule for the Azure connector is to have it run every 30 minutes. Coincidentally, this happens to be the same interval that IBM requires for hardware checks of devices linked to IBM PVU licenses (when you have IBM's approval to use FlexNet Manager Suite as the source for sub-capacity licensing calculations, for which see the *Sub-*

Capacity Licensing with IBM PVU chapter of the *FlexNet Manager Suite System Reference* PDF, available through the title page of online help).

However, each of these coincidental but distinct schedules has its own separate reasons:

- The Azure connector should run every 30 minute to allow it to capture transient information that Azure keeps available for only an hour, so that you can track your terminated instances where required. This is the *connector* schedule, and the connector does not return sufficient inventory detail to satisfy the IBM requirements.
- The hardware check for IBM PVU licenses must run every 30 minutes as part of your amended contractual agreement with IBM. This is an *inventory* schedule that controls operations of FlexNet inventory agent embedded on an operational instance.

If you are providing IBM PVU licenses for use in the Microsoft Azure cloud, *and* have the IBM amended agreement to allow sub-capacity calculations within FlexNet Manager Suite, these two default schedules conveniently satisfy both requirements within Azure. However, be aware of the following requirements:

- IBM PVU licenses in the BYOSL model are not appropriate for short-lived instances that are then terminated (see previous section about the pitfalls of terminating instances with BOYSL). Start/stop operations can be used when necessary, as long as the instance is maintained and not terminated.
- Persistent (unchanging) inventory device names and matching instance IDs are critical to this license model, allowing proper calculations of peak consumption values over time. For details about setting a unique but persistent device name for inventory, see Collecting Inventory from Instances.
- As always, full hardware and software inventory (from the instance) is required for IBM PVU licensing. Compliance with your IBM license requires regular software inventory (daily is recommended best practice) as well as the 30-minute hardware check. Use of the FlexNet inventory agent is also mandated by your adjusted license agreement. The techniques in Collecting Inventory from Instances prepare an Azure VM Image from which you can launch instances with the operational FlexNet inventory agent embedded, satisfying those requirements.
- As always, the first full software and hardware inventory must be uploaded and processed on the central application server *before* the 30-minute hardware checks take effect. (This allows the calculation of updated rule targets that identify all devices affected by IBM PVU licensing processes.) Typically the relevant compliance calculations take place overnight, after which the revised device policy must be distributed through the inventory beacon hierarchy to affected installations of FlexNet inventory agent, bringing with it the special schedule for the hardware check. All of this process means that in the normal course of events, IBM PVU operations typically are running effectively from the day *after* an instance is launched with PVU-licensed software in operation.

Uncapped calculations except on dedicated hosts

Some license types allow that, where the total license liabilities for many VMs on the same host theoretically exceed factors like the total number of cores available on the host, the summing process is 'capped' or given an upper limit by the actual capacity of the host. For example, if five VMs on a host are each allocated 2 cores, for a total of 10 allocated cores when the host actually only has 8 cores available, then, just as in reality the time-sharing between the VMs limits the cores in use at any moment, so also the license rules cap (or reduce) the calculated figure (10 cores) to the total capacity of the host (8 cores).

Obviously, capping of consumption calculations requires inventory not only of all the VMs on a host, but also of the host itself. However, in the case of Microsoft Azure instances, inventory details are not available for any host types other than 'dedicated hosts', over which you have full control. This means that on all other, non-dedicated hosts in Azure, capping of license calculations cannot occur.

All calculations for BOYSL licensing of instances on non-dedicated Azure hosts are based entirely on the inventory taken from the instances themselves, without capping. This does not expose you to any risk of under-licensing (provided that all instances are returning inventory); the only risk is that you may *perhaps* be over-calculating the consumption that *might* apply on a known host with capping applied.

However, this may not be an issue at all. Carefully check your BYOSL terms and use rights. You may find that software publishers already disallow capping in cloud environments, because of the impossibility of tracking instances that may pop up now on this host and now on another, depending on resource allocations within Azure.

3

Configuring an Azure VM Image for Instances

The easiest way to collect inventory from all your cloud-based instances is to ensure that they are launched from an Azure image that already contains everything needed for the collection of FlexNet inventory. The requirements include:

- Installing the latest approved FlexNet inventory agent into the image
- Adding a customized configuration file that identifies the upload location for inventory
- Installing a customized schedule to manage FlexNet inventory agent in the resulting instances, triggering inventory gathering on start-up (and best practice is to include a backup schedule for further inventory collection in case an instance from this AMI runs for an unexpectedly long time)
- Including the current version of `InventorySettings.xml` that provides advanced functionality for FlexNet inventory agent, since the short lifetime does not allow for normal policy-based downloads
- Ensuring a unique name for each instance created from this Azure VM Image.

At this summary level, the requirements are the same for Windows and Linux platforms. In the details, of course, there are platform specifics.

To prepare an Azure Resource Image:

1. In the web interface for FlexNet Manager Suite, navigate to **Discovery & Inventory > Inventory Settings**.

   The **Inventory Settings** page displays.

2. Expand the **Inventory agent for download** section.
3. Collect the template configuration file:
   - For a Windows-based AMI, click **Download bootstrapping template file**, and save `mgssetup.ini` to a convenient working folder (such as `C:\temp`). (Do not change the file name.)
   - For a Linux-based AMI, in the *Gathering FlexNet Inventory* PDF (available through the title page of online help), copy the text from *Agent third-party deployment: Sample UNIX Bootstrap Configuration File* and save it as `mgsft_rollout_response` in a convenient working folder (such as `C:\temp`).

4. From the **Inventory agent** drop-down list, select the version of FlexNet inventory agent you want to install in your AMI, and save it to your working folder.

   In general, install the latest available version, subject to your corporate policies. This provides access to the latest functionality. For example, to include advanced inventory for Microsoft Azure, you must use FlexNet inventory agent 13.2.0 or later (see Appendix 3: Enhanced Inventory Gathered by Agent).

5. In your preferred flat text editor, customize your bootstrapping template file to be used for FlexNet inventory agent in your Microsoft Azure environment as follows, saving the edited version in a separate subfolder.

   For Windows, the only *mandatory* change is to identify the upload location for gathered inventory, as described below. On Linux, FlexNet inventory agent requires both an upload location and a download location. As always, on either platform, experts may also customize other preferences needed for your implementation, as described in the platform-specific topics in the *Gathering FlexNet Inventory* PDF:

   - *Agent third-party deployment: Edit the Configuration File for Microsoft Windows*
   - *Agent third-party deployment: Configure the Bootstrap File for UNIX*.

   The upload (and download) locations require a URL to the host inventory beacon. Best practice is to provide a DNS alias for your chosen inventory beacon, so that when circumstances change, a simple switch of the alias leaves any running and future instances fully operational, without requiring changes to the virtual machine image.

   - For **Windows**, in `mgssetup.ini`:

     a. Locate this section for `Common` preferences, and uncomment (remove the leading semi-colon) and edit the following settings:

     ```
     ;=========================================================================
     ; Registry settings to be created under
     ; HKLM\Software\ManageSoft Corp\ManageSoft\Common
     [Common]
     desc0 = UploadSettings\Bootstrap Server\Protocol
     val0  = http
     desc1 = UploadSettings\Bootstrap Server\Priority
     val1  = 100
     desc2 = UploadSettings\Bootstrap Server\AutoPriority
     val2  = False
     desc3 = UploadSettings\Bootstrap Server\Host
     val3  = beacon.fnms.com
     desc4 = UploadSettings\Bootstrap Server\Port
     val4  = 80
     desc5 = UploadSettings\Bootstrap Server\Directory
     val5  = /ManageSoftRL/
     ```

     b. Optionally, modify the `Protocol` value if you want to use HTTPS (`val0 = https`).

     c. You *must* customize the `Host` preference (shown above as `val3 = beacon.fnms.com`) to your chosen DNS alias (or preferred host setting) and your own enterprise name if the inventory beacon is located in your enterprise network rather than on an Azure instance. For an inventory beacon hosted on an Azure instance, you can identify its public and private DNS hostnames on the Azure console (select **Instances** in the navigation pane, choose the instance hosting the inventory beacon, and read both its public and private DNS hostnames in the details pane).

> ***Tip:*** *If you have an inventory beacon on an Azure instance, your installations of FlexNet inventory agent on other instances can be configured to use the private DNS hostname for accessing the inventory beacon (provided that both are within the scope of your VPC). The private hostname has greater stability, particular if the inventory beacon host is stopped and started from time to time.*

d. Optionally, customize the port setting (for example, if you are switching to the HTTPS protocol, the default port is `443`).

> ***Important:*** *The `Directory` preference is mandatory, and must be set as shown above.*

e. Create a subfolder (such as `SysPrepAgent`) and save your edited `mgssetup.ini` there. Do not change the file name, which is mandatory.

- For **Linux**, in `mgsft_rollout_response`:

a. In the first section of the file, customize the setting for the download location (see also the comments for the upload location, next):

```
# The initial download location(s) for the installation.
# For example, http://myhost.mydomain.com/ManageSoftDL/
# Refer to the documentation for further details.
MGSFT_BOOTSTRAP_DOWNLOAD=http://beacon.fnms.com:8080/ManageSoftDL/
```

b. In the next section of the file, customize the setting for the reporting (upload) location:

```
# The initial reporting location(s) for the installation.
# For example, http://myhost.mydomain.com/ManageSoftRL/
# Refer to the documentation for further details.
MGSFT_BOOTSTRAP_UPLOAD=http://beacon.fnms.com:8080/ManageSoftRL/
```

You may customize the protocol to HTTPS if required, and omit or customize the port number as required. You *must* customize the host URL (shown above as `beacon.fnms.com`) to your chosen DNS alias (or preferred host setting) and your own enterprise name if the inventory beacon is located in your enterprise network rather than on an Azure instance. For an inventory beacon hosted on an Azure instance, you can identify its DNS hostname on the Azure console (select **Instances** in the navigation pane, choose the instance hosting the inventory beacon, and read its DNS hostname in the details pane).

> ***Important:*** *In each case, the trailing `/ManageSoftDL/` (download location) or `/ManageSoftRL/` (upload or reporting location) is mandatory, and must be specified as shown above.*

c. In the last line of the file, switch the value to prevent policy running after installation, as short-lived instances do not have time to wait for policy and its outcomes:

```
MGSFT_RUNPOLICY=0
```

d. Create a subfolder (such as `SysPrepAgent`) and save your edited `mgsft_rollout_response` there. Do not change the file name, which is mandatory.

e. Unless you already have a strategy and practice that gives every Linux instance a unique name, also create a new text file for your Linux virtual machine image, called `tmpconfig.ini`, with exactly the following two

lines:

```
[ManageSoft\Common]
MachineID=
```

This file is deliberately incomplete at this stage, and is to be completed at instance start-up, as described later.

6. For either platform, copy the following and paste it into a plain ASCII text file, and save it in your `SysPrepAgent` folder as `DefaultSchedule.nds` (this file name is mandatory). Of course, you may choose to unwrap those lines wrapped for readability here.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Schedule PUBLIC "Flexera Inventory Manager schedule"
     "http://www.managesoft.com/dtd/schedule.dtd">

<Schedule NAME="Default Schedule" TIMEDATESTAMP="20180903T044104Z"
SCHEDSCHEMA="60">
        <Event NETWORK="False"
                 NAME="Generate Inventory"
                 CATCHUP="Never" ID="{F065A477-B500-4611-8340-3BF313FB785B}">
            <LogicalCommand PARAM=" -o UserInteractionLevel=Quiet"
                 ACTION="Report" COMPONENT="Tracker"/>
           <Trigger TYPE_PARAM="0" TIMESTART="143950"
                 DATESTART="20180903" TYPE="Logon" MAXDELAY="0"/>
        </Event>
      <!-- Following two events are optional (see notes) --!>
        <Event NETWORK="False"
                 NAME="Generate Inventory" IDLEDURATION="86400"
                 CATCHUP="Never" ID="{F065A477-B500-4611-8340-3BF313FB785B}">
            <LogicalCommand PARAM=" -o UserInteractionLevel=Quiet"
                 ACTION="Report" COMPONENT="Tracker"/>
            <Trigger TYPE_PARAM="1" TIMESTART="001000"
                 DATESTART="20180903" TYPE="Daily" REPEAT="21600"
                 TIMEWINDOW="3600" DURATION="86400"/>
        </Event>
       <Event NETWORK="False"
                 NAME="Upload Client Files"
                 CATCHUP="Never" ID="{9CB87BAE-2829-498C-8643-10A9BD3BFA56}">
            <LogicalCommand PARAM="-a"
                 ACTION="Upload" COMPONENT="Uploader"/>
            <Trigger TYPE_PARAM="1" TIMESTART="002000"
                 DATESTART="20180903" TYPE="Daily" REPEAT="600"
                 DURATION="86400"/>
        </Event>
</Schedule>
```

The first event in this schedule triggers machine inventory collection on startup (`TYPE="Logon"`), and the `tracker` component automatically attempts an upload to the bootstrap inventory beacon as soon as inventory gathering is complete. If you are certain that all instances from this image are short-lived, only this event is

required. If you are using an instance, and you choose to stop and restart your instance (rather than to terminate and re-launch it), this inventory is checked on each restart.

In addition, two further schedule events provide easy management of cases where an instance proposed to have a short life in fact keeps on running for some longer time.

The second event also generates inventory, and is triggered every 6 hours (`REPEAT="21600"`, with all times expressed in seconds), to start within an hour thereafter (`TIMEWINDOW="3600"`). However, this inventory collection is different than the first case, because here the FlexNet inventory agent does not repeat the inventory collection if it has been successful within the last 24 hours (`IDLEDURATION="86400"`). This means that, effectively, the FlexNet inventory agent checks four times a day whether there has been a successful inventory collection in the last 24 hours, and if not, it runs a new inventory collection.

In a perfect world, the third event is redundant because the `tracker` component uploads inventory as soon as it is collected. This third, separate upload event is a catch-up to work around transient network issues. It checks every 10 minutes to see whether any inventory files have failed to upload (and are therefore still waiting in the staging folder). In normal cases, nothing is waiting, and the `uploader` shuts down immediately, causing negligible load on the instance. But where there have been networking issues, upload is attempted again every 10 minutes until all files are successfully uploaded and removed from the staging folder.

7. On a convenient inventory beacon, navigate to `%CommonAppData%\Flexera Software\Staging\Common\ClientConfiguration`, take a copy of `InventorySettings.xml`, and save it to your `SysPrepAgent` working folder.

   This completes the preparation of materials for your virtual machine image for instances of this expected life. Now continue to use these materials to prepare the virtual machine image. For both platforms, the high-level process is to customize a running instance, and then save it as a virtual machine image.

   For a Windows-based VM Image please refer to the following link:

   - https://docs.microsoft.com/en-us/azure/virtual-machines/windows/tutorial-custom-images

   For a Linux-based VM Image please refer to the following link:

   - https://docs.microsoft.com/en-us/azure/virtual-machines/linux/tutorial-custom-images

8. Following the Azure documentation, select (or create), launch and connect to your chosen instance from which you are creating your Azure VM Image.

9. To customize your instance:

   a. Ensure that the destination folder exists for your installation of FlexNet inventory agent, and if not create it.

      The default paths for each platform are:

      - On Windows, `C:\Program Files (x86)\ManageSoft`
      - On Linux, `/opt/managesoft/bin`.

   b. From your local `SysPrepAgent` working folder, copy these files into the destination folder on your instance:

      - `DefaultSchedule.nds`
      - `InventorySettings.xml`
      - Only on Windows, `mgssetup.ini`

- Only on Linux, `tmpconfig.ini`.

c. For Linux only, also copy your amended `mgsft_rollout_response` file to `/var/tmp/` on your instance.

d. Collect the downloaded installer for FlexNet inventory agent from your working folder, and run the installer on your instance, installing FlexNet inventory agent into the prepared destination folder.

The installer automatically configures the custom settings declared in the configuration file (either `mgssetup.ini` or `mgsft_rollout_response`, depending on platform). For more information about running the installation, see the *Agent third-party deployment: Details* chapter of the *Gathering FlexNet Inventory* PDF, focusing on the installation topic for your preferred platform.

e. When installation is complete, in the same destination folder and for both platforms, run the following command to install your default schedule:

```
start ndschedag.exe -t machine .\DefaultSchedule.nds
```

This virtual machine image is now ready. As soon as each instance is spawned, the embedded FlexNet inventory agent gathers inventory from the instance, and immediately uploads it to the bootstrap inventory beacon. Depending on the resources on each instance, this process may take around 2-5 minutes, after which the instance can be stopped (but remember, not terminated if you wish to track related software licenses in FlexNet Manager Suite).

4

Appendices: Technical Data

The following topics do not include content you need for set-up or operation of the Microsoft Azure connector. These are the deep-down technical data that you can ignore until you have a real need.

Appendix 1: Cmdlets in Azure PowerShell

The connection from FlexNet Manager Suite to the Microsoft Azure service relies on the AzureRM Tools for Windows PowerShell (as described in Prerequisites and Setting Up). Within that toolset, the connector uses the following PowerShell cmdlets:

- `Connect-AzureRmAccount` — To log into Microsoft Azure
- `Disconnect-AzureRmAccount` — To log out Microsoft Azure
- `Get-AzureRmVM` — To return the list of the instances (virtual machines) you own, and then collect summary inventory details on each one
- `Get-AzureRmLocation` — To identify the available Azure geographic regions of the instance type (size in Microsoft term) where your instances may be running
- `Get-AzureRmVMSize` — To return the list of the instances type based on location.

Appendix 2: Data Imported by Microsoft Azure Connector

The PowerShell connector for Microsoft Azure returns information about each Azure instance, and its host. All instances are subsequently displayed in the **Cloud Service Provider Inventory** page. Hosts are never displayed in this page, as in general hosting is variable and controlled by Microsoft Azure.

Tip: *One exception where you have direct control is the 'dedicated host'. Like all other host types, dedicated hosts do not appear in the* ***Cloud Service Provider Inventory*** *listing; but they do automatically appear in inventory device listings, such as the* ***All Inventory*** *page, where you can search for them using the* ***Hosted in*** *property. Any instances*

*from the **Cloud Service Provider Inventory** listing that are hosted on your dedicated host also display a link to the dedicated host properties in the **Host** column.*

Instance data

***Tip:** All kinds of instances return this set of data, whether they are virtual machines or bare metal instances. The latter return an `Instance type` starting with `i3.`.*

If the data gathered through the connector matches inventory collected from another source, the **Cloud Service Provider Inventory** page includes a hyperlink to open the inventory device properties page for the instance. Putting that another way, inventory device records are *not* created when the Azure connector is the sole source of information; but they *are* created when inventory is returned from another source that covers the cloud instance (one possible example is having FlexNet inventory agent running locally on the instance). In this latter case, linked records exist in both the `ComplianceComputer` and `CloudServiceInstance` tables.

Some data is additional to the normal content collected as inventory. For example, the `InstanceTenancyID`, used for identifying whether the current device is a `Dedicated instance`, or is an instance running `On dedicated host`, is imported into the `CloudServiceInstance` table, and, as **Availability type**, is visible in both the **Cloud Service Provider Inventory** page and the **Cloud Hosting** tab of the inventory device properties (where inventory exists for the instance).

***Tip:** Azure "bare metal instances" behave like other instances in terms of data gathered and records created.*

When there is an inventory device record for an instance, the inventory-related values imported through the Microsoft Azure connector (and shown below in alphabetical order) are displayed in the separate **Cloud Hosting** tab of the inventory device properties:

- `AvailabilityZone` (state of the instance, will be mapped to `Started`, `Stopped`, or `Unknown`)
- `CoreCount`
- `ImageID`
- `InstanceCloudID` (Azure VmId, which is also UUID of the VM)
- `InstanceType`
- `Region`
- `VMEnabledStateID`.

Appendix 3: Enhanced Inventory Gathered by Agent

As well as the high-level data gathered from the Microsoft Azure connection, FlexNet inventory agent (version 13.1 or later) gathers addition inventory data for each Azure instance where FlexNet inventory agent is locally installed. To do so, FlexNet inventory agent access the following Azure-internal URL:

- `curl -H Metadata:true "http://169.254.169.254/metadata/instance?api-version=YYYY-MM-DD"`

— you must replace YYYY-MM-DD with the api-version date to return the unique identifier for the Azure instance. For example, `curl -H Metadata:true "http://169.254.169.254/metadata/instance?api-version=2018-10-01"`.

- To view the api-version dates enter `curl -H Metadata:true "http://169.254.169.254/metadata/instance"` which will return an error message which identifies the api-versions. For example:

```
{
"error": "Bad request. api-version was not specified in the request. For more
information refer to aka.ms/azureimds", "newest-versions": [
"2018-10-01",
"2018-04-02",
"2018-02-01"
]
}
```

- `http://169.254.169.254/metadata/instance` returns a JSON-formatted value containing data about the instance where FlexNet inventory agent is locally installed. For example:

```
{
    "devpayProductCodes" : null,
    "marketplaceProductCodes" : [ "1abc2defghijklm3nopqrs4tu" ],
    "availabilityZone" : "us-west-2b",
    "privateIp" : "10.158.112.84",
    "version" : "2017-09-30",
    "instanceId" : "i-1234567890abcdef0",
    "billingProducts" : null,
    "instanceType" : "t2.micro",
    "accountId" : "123456789012",
    "imageId" : "ami-5fb8c835",
    "pendingTime" : "2018-11-19T16:32:11Z",
    "architecture" : "x86_64",
    "kernelId" : null,
    "ramdiskId" : null,
    "region" : "us-west-2"
}
```

These two data points are then embedded within the normal `.ndi` inventory file as a pseudo-hardware WMI class called `MGS_CloudMetadata`. The class looks similar to this example (lines wrapped for presentation):

```
<Hardware class="MGS_CloudMetadata" Name="ec2amaz-vrkiu0p" Evidence="metadata">
    <Property Name="instance-id" Value="i-1234567890abcdef0"
              Evidence="/latest/meta-data/instance-id"/>
    <Property Name="document" Value="ewogICJwcml2YXRlSXAiIDogIjE3Mi4zMS4yLjExNiIs...."
              Evidence="/latest/dynamic/instance-identity/document"/>
</Hardware>
```

After a full inventory import, these details are eventually saved in the `CloudServiceInstance` table in the compliance database.

VI

Data Platform Integration

This part introduces the integration between FlexNet Manager Suite and Data Platform, an umbrella product that includes both Technopedia and Normalize.

The method of integration is a straight-forward connector, available without special installation on your inventory beacons (including any inventory beacon you may have installed on your central application server). On the Data Platform side, an enhanced license term and a new data pack are required (see Configuring the Data Platform Connector for details).

Such simplicity means that the main use for the documentation in this part is to understand the mapping between

- Database tables and columns that underlie Normalize, and
- The destination tables and columns in the compliance database for FlexNet Manager Suite.

Also important is the embedded commentary on the current limitations of Data Platform as a *sole* data source driving license compliance calculations. When you have multiple inventory sources, imports from Data Platform can be a valuable addition to your data gathering.

Supported version

The Data Platform v5 connector supports import only from version 5 of Normalize.

1

Integration with Data Platform v5

Data Platform version 5 (formerly from BDNA, and now from Flexera) combines Technopedia, a reference catalog with 120 million datapoints across more than 1.7m products, and Normalize, which provides a unified view of the data applicable to your environment.

To integrate this data for your use in license and asset management within FlexNet Manager Suite, release 2019 R2 includes a connector that imports normalized data from Normalize v5 into your compliance database. The imported data includes:

- Records of inventory devices (for both physical servers and virtual machines) found in your environment by the inventory tools (such as Microsoft SCCM or BMC ADDM) that are feeding into Normalize v5
- Details of applications recognized within your enterprise through those tools
- Installation records that link the found software to the inventory devices where it is installed
- Details of end-users who log onto the inventory devices
- Where available, records of application usage (how often, and for how long, an end-user utilized an application).

Application records are dealt with in the same way as inventory data from any other source. This means that the data representing applications within Normalize v5 is imported into FlexNet Manager Suite in the first place as inventory evidence (in this case, as installer evidence). What happens next depends on the details of the software:

- If the application is already known to the Application Recognition Library (ARL), and therefore has a Flexera ID included in the import from Data Platform, the evidence is recognized by the ARL and a matching installed application record is created (if it did not exist already) and displayed in the **Installed Applications** page. For such records, the inventory evidence type is set to `FlexeraID`, and the installer evidence record is linked to the application record.

  Tip: *It may happen that the application is only recognized by a later update of the ARL than is currently deployed in your enterprise. If this occurs, a widget on the* ***System Health Dashboard*** *highlights the need to refresh your local, downloaded ARL (or you can wait until the next automated download, which typically happens each weekend).*

- If the installer evidence is not [yet] recognized by the ARL, then following normal system processes, it cannot be linked to an application record, and it remains in the **Unrecognized Evidence** page, in the listing under the **Installer evidence** tab. For these cases, the inventory evidence type is displayed as `Data Platform`. (Future updates to Data Platform and the ARL may mean that the same installer evidence re-imported thereafter may be recognized, and an installed application record automatically created at that time. For this reason, it is best not to manually trigger

application creation from such records of *public* applications, but to wait for Flexera's continuing alignment work to automatically complete the links for you. If you find such a case that is needed in your enterprise with some urgency, you can ask Flexera to prioritize its delivery through a later release of the ARL.)

Tip: *A subset of these unrecognized cases may be application records that you created locally within your enterprise (records that in Data Platform are called "private applications", and in FlexNet Manager Suite are called "local applications"). As such records may never be matched by the ARL, you may wish to start from the unrecognized evidence record to re-create your local application record within FlexNet Manager Suite. In summary, best practice is to manually trigger the creation of application records only for locally-created, 'private' applications; and to await updates from Flexera for 'public' applications.*

Connector, not adapter

The integration with Data Platform v5 is not strictly an "adapter", for the following reasons:

- It does not require any custom XML file to be maintained through either the Inventory Adapter Studio or the Business Adapter Studio
- There is no need (nor purpose) for any staging database
- There is no separate download and installation required (with the possible exception of the initial release, where shipment of the connector is not synchronized with the product release; but in future it is embedded as a standard part of the product).

Instead, this integration is through a *connector* that is available on any up-to-date inventory beacon. You set up the import in exactly the same way as for any other inventory connection: specifying the details in the new inventory connection dialog on the **Inventory systems** page on your preferred inventory beacon; and nominating your preferred schedule for the regular imports. Full details are included below in Configuring the Data Platform Connector.

Inventory gaps and limitations

Following topics cover the mapping of data from within Data Platform to the import tables within the compliance database in FlexNet Manager Suite. (Hints are also given for tracking data from the import tables to the operational tables within the compliance database, with further information being available in the *FlexNet Manager Suite Schema Reference* PDF, available through the title page of online help.) Within those topics are specific comments about the impact of particular inventory gaps on tracking your license compliance.

At a high level, inventory gathered *only* from Normalize v5 is insufficient for:

- Oracle Processor license calculations (missing the partial number of processors)
- IBM PVU license calculations (missing the partial number of processors)
- Any licenses that rely on details about the host settings for virtual machines, such as *assigned* core or processor counts (for VMs, this connector can only import the virtual host, and no other details of the VM, including pools, clusters, capping and the like). Notice, however, that inventory is normally gathered separately both from the virtualization host and from available guest machines, so that records are available for each device's view of its *own* properties.

Validate your data processes

When there are inventory gaps, the best-practice work-around is to add a second inventory source that fills the gaps, and

allow FlexNet Manager Suite to merge the sources to produce a unified data set. The risk factor is that record merging may fail (because of different values in key fields), leading to duplicate records in your compliance database that cause over-counting.

Such a case is the possible merging of imports from Data Platform with another source, such as the FlexNet inventory agent. One of the key fields for merging inventory device records is the domain name that the device reports, and currently the values for domain are by default saved differently. Data Platform v5 typically saves a flat domain name (such as `flexera`); and while FlexNet Manager Suite can handle either format, by default is saves and uses the DNS domain name format (distinguished name) that includes the root domain (such as `flexera.com`). At first glance this difference looks high-risk; but there is a compliance database table that helps to resolve this issue, called the `Domain` table. Since this stores both the fully-qualified distinguished name for the domain, and also its flat name, it allows reliable matching between the two forms, *provided that* it has been populated with the appropriate records before a mapping is requested.

So, to prevent duplicate inventory device records, it is important to ensure that the `Domain` table is populated with all your domain names in both formats before importing from Data Platform (or indeed, any other inventory source that reports the flat name for domains). There are three ways to do this, of which the first is best practice:

- Run an import from Active Directory. This populates the `Domain` table appropriately with both domain name formats.
- Import FlexNet inventory, since the FlexNet inventory agent reports both name formats.
- Import inventory from Microsoft SCCM 2012 or later, since from that release, SCCM also reported the domain name in both formats.

The first option is clearly preferable since it gives the most complete coverage of all your domains. Therefore, before importing from Data Platform, ensure that your Active Directory imports are up-to-date.

Tip: *If you have domains that are outside the reach of your Active Directory implementation, you can create a spreadsheet and write a business adapter to populate the `Domain` table.*

A match between host name, domain, and serial number is sufficient for merging (de-duplicating) inventory device records. Where these high-priority values are not available, FlexNet Manager Suite continues checking lower-priority properties including:

- `HostIdentifyingNumber`
- `HostType`
- `ILMTAgentID`.

Since all these properties are unavailable from Data Platform v5 (see Imported Computers (Inventory Devices)), this makes correct mapping between the naming formats of your domains all the more important in avoiding duplicate device records.

Attention: *Even when you follow best practice and use the above techniques to populate the `Domain` table with both the flat name and the distinguished name for each domain, issues with domain reporting may still cause creation of duplicate records for computers or users. This is because some data sources (notably HP-UD and ADDM) report domain properties for users and computers in ambiguous ways, and where overlapping data comes from another source (including coming through Data Platform even from the same original source), duplicate records may result. You may both prevent and repair such duplication simply by setting your* ***Primary*** *inventory source to a known-good choice from the overlapping sources, such as SCCM, FlexNet inventory, or Data Platform. (To find this setting, navigate to the*

*system menu (⚙ ▼ in the top right corner), choose **Data Inputs** and select the **Inventory Data** tab). To clear duplicate records previously created through this issue, set your good source that's reporting the domain as primary, and wait for (or trigger) a full import and compliance calculation. By default, these full imports and calculations happen overnight, so you can check that your previous duplicates are cleaned up the day after adjusting your primary inventory source.*

If you are considering the use of two data paths for overlapping inventory records (such as having a common source imported directly into FlexNet Manager Suite on the one hand, and imported through Data Platform v5 on the other), one caveat to be aware of is the need for additional expertise. Suppose that you are trying to analyze a problem with recognition of a particular application. With two data paths, you must first determine which data path delivered the record, because the recognition logic differs between the paths:

- If the path was direct from the inventory source (such as SCCM) into FlexNet Manager Suite, then the evidence rules in the Application Recognition Library (ARL) must be assessed for effective matching.
- If that path from the inventory source led through Data Platform v5, you first need expertise to analyze the recognition process within that domain, because it is Data Platform (and in particular Normalize v5) that has recognized the application. In addition, you must validate that Data Platform v5 is supplying a Flexera ID for the application it recognizes, and that the same Flexera ID is listed as evidence in your installed ARL — otherwise, despite its initial recognition, the 'application' languishes in the **Unrecognized Evidence** page within FlexNet Manager Suite.

In summary, combining both data paths provides a growth opportunity.

Supported versions

The connector is validated for operations between Data Platform v5 and FlexNet Manager Suite 2018 R1. Within that framework, the following recommendations apply:

- **Data Platform:** The preferred release of Data Platform (and in particular, Normalize) is 5.5.4 (or later), as this release fixed a problem where, for virtual machines, imported data contained incorrect values for the count of **Cores** and **Threads** (or logical processors). Both these values are visible in FlexNet Manager Suite on the **Hardware** tab of the inventory device properties. In the same location, you may also manually override both these values when a correction is needed, and your overridden values are unaffected by future inventory imports (for more details, see the online help for the **Hardware** tab).
- **FlexNet Manager Suite:** While formal validation is limited to release 2018 R1 and later, informal spot testing suggests backward compatibility of the connector should extend back to at least release 2017 R1 of FlexNet Manager Suite.

Important: *If you are testing with an earlier version of FlexNet Manager Suite, be sure that, on all relevant inventory beacons, you remove the superseded folder `C:\ProgramData\Flexera Software\Compliance\ImportProcedures\Inventory\Reader\BDNA` before distributing the new `Data Platform v5` folder as a replacement (full details about the new folder are included in Configuring the Data Platform Connector).*

2

Configuring the Data Platform Connector

Configuring the Data Platform connector is a one-time task, and quite straight-forward. Thereafter, the imports continue at the intervals you schedule. Have ready your connection details for the Normalize v5 database.

To configure the Data Platform connector:

1. To minimize the risk of duplicate records when you have imports from multiple inventory sources, ensure you have completed Active Directory imports from all your domains.

   For more details, refer back to Integration with Data Platform v5. For guidance about Active Directory imports, see *FlexNet Manager Suite Help > Inventory Beacons > Active Directory Page*.

2. If necessary, apply for your updated Data Platform license to enable integration with FlexNet Manager Suite, by contacting your Flexera Support consultant.

   This updated license is available free of charge to existing customers who have implemented Data Platform. It authorizes additional functionality and a new data pack. If you are unsure whether you already have this updated license, there are two ways you may check:

   - You can send a question to Data Platform Support
   - You can run the following SQL statement against your Normalize v5 database:

     ```
     Select IS_SUBSCRIBED from BDNA_CP where CP_NAME = ‘Technopedia Flexera Mapping’
     ```

     If the result returns Y, your Data Platform instance includes the license term, and you may skip ahead to step 4.

     If it returns N, you need to apply for the license update.

3. When you receive the updated activation key:

   a. In BDNA Admin, navigate to the **Preferences** page.

   b. Enure that the **Registration** tab is selected (top left).

   c. Delete the current contents of the **Activation Key** field.

   d. Copy the key in full from the email you received, and paste it into the **Activation Key** field.

   e. Click **Register New Key**.

4. Download the new data pack:

   a. Navigate to the **Technopedia** page in BDNA Admin.

   b. Click **Start Catalog Sync**, and wait for the data pack to download.

5. Ensure that the Data Platform connector is in place in your FlexNet Manager Suite implementation.

   You can check whether the Data Platform connector is already in place by inspecting any inventory beacon, or (for an on-premises implementation) your central application server (in a multi-server implementation, check the batch server). In each case, open Windows Explorer and navigate to `C:\ProgramData\Flexera Software\Compliance\ImportProcedures\Inventory\Reader`. If you find a folder there called `Data Platform v5`, your connector is already in place, and you may skip ahead to the next major step. If not, switch to a web browser to obtain your copy:

   a. Access https://flexeracommunity.force.com/customer/articles/en_US/INFO/Adapter-Tools-for-FlexNet-Manager-Suite.

   Tip: *Access requires your Customer Community user name and password. If you do not have one, use the link on the login page to request one.*

   b. Click the link `Adapter Tools for FlexNet Manager Suite`.

      A new browser tab may appear temporarily, and the download of `Adapter Tools for` FlexNet Manager Suite `2019` R2`.zip` commences.

   c. In your browser dialog, choose to save the file, and if the browser allows it, direct the saved file to a convenient working location (such as `C:\temp` on a central, accessible server).

      If your browser saves the file to a default location (such as your `Downloads` folder), move or copy it to the appropriate working location when the download is finished.

   d. Right-click the downloaded zip archive, and choose **Extract All...**.

   e. Navigate through the unzipped archive to `Adapter Tools for` FlexNet Manager Suite `2019` R2`.zip` > `Data Platform v5`.

   f. Copy the `Data Platform v5` folder in its entirety to `C:\ProgramData\Flexera Software\Compliance\ImportProcedures\Inventory\Reader` on your chosen inventory beacon.

   Tip: *Copy the folder to the same location on your central batch server (or server hosting that functionality), and FlexNet Manager Suite automatically transmits it to all your inventory beacons as part of their regular policy update. Allow an hour or two (depending on how many inventory beacons you have) for this process to complete.*

6. Ensure that you have your preferred schedule for imports from Data Platform set on the appropriate inventory beacon:

   a. Log into the inventory beacon interface as an administrator (for example, in the Windows Start menu, search for **FlexNet Beacon**, right-click it, and select **Run as administrator**).

   Tip: *Remember that you must run the inventory beacon software with administrator privileges.*

b. From the **Data collection** group in the navigation bar, choose **Scheduling**.

c. If there is not already a suitable schedule in the list, click **New...** and complete the details (see the online help for that page for more information). Otherwise, identify the schedule you will use.

7. Define the connection to your Data Platform implementation:

   a. Select the **Inventory systems** page (in the same group), and then click **New...**.

      If you used the down arrow on the split button, use only the `SQL Server` option.

   b. Complete the details, using a name that is distinct within the first few characters so that it is recognizable in a narrow column.

      Important: *Take care that:*

      - *The **Source Type** is `Data Platform v5`*
      - *You provide the connection details for your Normalize database (not the Technopedia one).*

      See the help for this page for more details. When you are done, click **Save**.

8. Select your new connection from the displayed list, and click **Schedule...**.

9. In the dialog that appears, select the name of your chosen schedule for inventory collection through this connection, and click **OK**.

10. At the bottom of the **FlexNet Beacon** interface, click **Save**, and if you are done, also click **Exit**.

Tip: *Consider whether you want to select your connection, and click **Execute Now**, before you exit.*

The connection is now configured, and imports data from Data Platform on your chosen schedule. Shortly after each import is completed, the inventory beacon automatically uploads the resulting inventory to the central application server. After the next full import and compliance calculation (typically scheduled overnight), the inventory is visible in the web interface of FlexNet Manager Suite.

3

Data Mappings, Gaps, and Impacts

The following topics provide a mapping between the source tables (and their columns) in the Normalize v5 database, and the corresponding destination tables and columns in the compliance database within FlexNet Manager Suite. This mapping covers only the first stage of the process, into the tables with names starting `Imported...`. These are effectively staging tables that hold the data between the upload from Data Platform, and the major transition (or import) into the operational tables in the compliance database, normally followed immediately by the license compliance calculations (the calculations of consumption against entitlements). The descriptions also identify the operational tables where the data eventually resides, and for a more detailed understanding of those tables and their columns, see the *FlexNet Manager Suite Schema Reference* PDF, available through the title page of online help.

The topics are separated by the target `Imported...` tables, each of which is aligned with a particular data object within FlexNet Manager Suite. Within each listing, the columns are in the same order that they appear in the *Schema Reference*, for your convenience if you are cross-referencing both documents.

As well as the list of imported data, a second list within each topic identifies the values that *cannot* be imported through Data Platform v5, together with notes on the impact that these gaps have on compliance calculations. Some gaps affect the suitability of this data source when you are managing certain types of licenses.

Each topic lists *every* column within the relevant `Imported...` table, so that if you are working across both this and the *Schema Reference*, there is no ambiguity about any column: it is either listed in the imports, or in the list of items missing from the import. Notice that this means some columns included for completeness appear in the list of "missing" items when in fact there is no expectation that they should be imported from any inventory source (or, to put it another way, they are *correctly* omitted from the import). These items are marked as "internally generated" by FlexNet Manager Suite.

Exempted kinds of data

The following kinds of data are *not* imported from Data Platform v5 to FlexNet Manager Suite:

- File evidence — this evidence type is not needed from Data Platform, as the import of installer evidence includes all required information
- WMI evidence — this evidence type is not needed from Data Platform, as the import of installer evidence includes all required information
- Client access evidence (used for Microsoft Client Access Licenses, or CALs) — this evidence is not available in Data Platform v5
- App-V evidence — this evidence is not available in Data Platform v5, and App-V application usage cannot be tracked if

Data Platform v5 is the single inventory source

- Oracle Database and Oracle option evidence, and Oracle LMS evidence — this evidence is not imported from Data Platform v5, so that recognition of Oracle Database version and edition, or of Oracle options, is not possible if Data Platform v5 is the single inventory source (but this information, along with detailed content to deliver to Oracle License Management Service, is fully available within FlexNet Manager Suite using the FlexNet inventory agent).

Comparison of Sources

This topic has two parts:

- A comparison of the inventory sources that are supported by FlexNet Manager Suite and Data Platform v5
- For data sources fully supported by both, a summary of the inventory gaps (some of which may affect compliance calculations) with data imported from each source either directly into FlexNet Manager Suite, or firstly into Data Platform and then imported from Data Platform v5 into FlexNet Manager Suite.

For additional detail, down to the level of individual data fields, refer to subsequent topics.

Supported inventory sources

For each of the data sources (listed alphabetically), this table displays:

- "Y" when either product offers out-of-the-box support for imports from that source
- A blank in either product's column when the data source is not supported
- "C" when either product has a *custom* extractor/adapter available to collect data from the source and import it into the connecting product.

Tip: *Support in FlexNet Manager Suite may be offered through a connector, requiring minimal configuration other than identifying the source; or an adapter, which may require a separate download and installation, and possibly setting up a staging database. In addition, FlexNet Manager Suite supports the development of custom adapters, for example, through the Inventory Adapter Studio. (For details, see the FlexNet Manager Suite System Reference PDF, available through the title page of the help.) For Data Platform, the equivalent terminology is extractor.*

Inventory source	FlexNet Manager Suite	Data Platform v5
Symantec IT Management Suite (formerly Altiris)	Y	C
App-V Standalone	Y	
BMC BladeLogic Client Automation (previously Marimba)	Y	C
BMC Discovery (previously ADDM)	Y	Y
EdgeSight for Citrix Virtual Apps (previously XenApp EdgeSight)	Y	
Citrix XenApp server agent (for Citrix Virtual Apps)	Y	
FlexNet Inventory Manager (previously ManageSoft)	Y	

Inventory source	FlexNet Manager Suite	Data Platform v5
FlexNet Manager for Engineering Applications	Y	
HP Discovery and Dependency Mapping Inventory (DDMI)	Y	C
HPE Universal Discovery (HP-UD)	Y	Y
IBM BigFix Platform (previously Tivoli Endpoint Manager)	Y	Y
IBM License Metric Tool (ILMT) or IBM BigFix Inventory (see note 1)	Y	
JAMF Pro (previously Casper)	C	Y
Microsoft Exchange ActiveSync	Y	
Microsoft Office 365	Y	
Microsoft SCCM (previously SMS)	Y	Y
Salesforce	Y	
SAP (through FlexNet Manager for SAP Applications)	Y	
ServiceNow	(see note 2)	Y
SolarWinds Orion	C	Y
Symantec IT Management Suite (formerly Altiris)	Y	C
Tanium Asset	C	Y

Note: *In the table above:*

1. *For imports from IBM BigFix Inventory, only applications with `IBM` as the `Publisher` are recognized by the Application Recognition Library.*
2. *FlexNet Manager Suite does provide for integration with ServiceNow, but in this case it is not importing raw inventory. ServiceNow is regarded as a source of business data rather than inventory data for FlexNet Manager Suite.*

Inventory gaps by import path

The following four inventory sources (column 1) are fully supported by both FlexNet Manager Suite and Data Platform v5. Based on the import path, there may be gaps in the imported inventory, and these gaps may be significant enough to prevent compliance calculations for related license types:

- The middle column identifies inventory gaps when FlexNet Manager Suite imports inventory data *directly* from each source
- The final column identifies inventory gaps if the inventory is first imported into Normalize v5 (within Data Platform), and subsequently imported from there into FlexNet Manager Suite.

Data source	Direct import to FlexNet Manager Suite	Import through Data Platform
BMC Discovery (previously ADDM)	No inventory gaps.	All virtualization data is missing.
HPE Universal Discovery (HP-UD)	No inventory gaps.	Advanced virtualization data missing *(see note below)*. DNS domain name missing (domain flat name is available).
IBM BigFix (previously Tivoli Endpoint Manager)	All virtualization data is missing.	All virtualization data is missing.
Microsoft SCCM (previously SMS)	All virtualization data is missing.	Advanced virtualization data missing. DNS domain name missing (domain flat name is available).

Note: *This table distinguishes between:*

- *Simple virtualization calculations for technologies like VMware ESX and Hyper-V — these require only the host/guest relationships to be tracked*
- *Advanced virtualization calculations for hardware partitioning technologies like HP-UX vPar, IBM LPAR, and Solaris zones — these require tracking of greater detail, such as partial processors and processor pool information. Information about clusters and host affinity (and the like) is also included in the advanced class.*

And obviously, "all virtualization data" includes both of the above.

Imported Installer Evidence

The `ImportedInstallerEvidence` table is a staging table used for input of records about installation evidence. From here, de-duplicated and normalized records are merged into the `InstallerEvidence` table.

`InstallerEvidence` typically lists installer evidence left behind after a particular item of software has been installed on a computer. In the case of imports from Data Platform v5, applications identified there are first imported as installer evidence. As always, this installer evidence is tested against rules saved in the Application Recognition Library (ARL), and when matched, the appropriate application record is added to the `InstalledApplications` table (if it does not already exist there), and is linked to the installer evidence. This application record is then visible in the **Installed Applications** page of the web interface for FlexNet Manager Suite. A record of installer evidence that cannot be matched by the ARL appears in the web interface on the **Unrecognized Evidence** page.

This process means there are three cases to consider, any of which may produce inventory gaps that can impact compliance calculations:

1. For applications recognized within Data Platform, and for which the normalized installer evidence has been mapped to an entry in the Application Recognition Library (ARL): there is no inventory gap as such, since at a minimum all these items are recognized within FlexNet Manager Suite; but in a few cases, some application variants (such as interface language, hotfix number, or platform) may be recognized by either tool but not the other.

2. For applications recognized within Data Platform, for which there is as yet no mapping to the ARL: the unrecognized evidence in FlexNet Manager Suite may mean that license consumption is under-counted because the applications have not been recognized within FlexNet Manager Suite, and therefore the licenses cannot measure consumption against the 'missing' applications.
3. For applications not recognized within Data Platform, but which could have been recognized by the ARL from other inventory sources: where Data Platform is your *only* data source, the unrecognized application may result in under-counting of license consumption.

There are two listings below. The first maps what is imported from Data Platform v5. The second lists values that FlexNet Manager Suite expects to load into the `ImportedInstallerEvidence` table, that are not available from Data Platform v5.

Data mapping for imported installer evidence

The following listing matches the source data from Data Platform v5 with the equivalent column in the `ImportedInstallerEvidence` table. Other columns from the `ImportedInstallerEvidence` table cannot be populated from Data Platform v5, and so are omitted here (and shown below). For more details on the database tables and columns within FlexNet Manager Suite, please see the *FlexNet Manager Suite Schema Reference* PDF, available through the title page of online help.

Data Platform (Table)/Column	FNMS (Table)/Column	Notes
(`MATCH_HOST_SW_PROD`) / `CAT_SW_UUID`	(`ImportedInstallerEvidence`) / `ExternalInstallerID`	The identifier used in Data Platform for the installer evidence.
(`MATCH_HOST_SW_PROD`) / `CAT_PRODUCT_NAME` *or* the Flexera ID for the application if available in `ARL_ID` ***Tip:*** *A prefix of `arl://` is prepended to the `ARL_ID` on import.*	(`ImportedInstallerEvidence`) / `DisplayName`	The display name of the software recorded in Data Platform, which is imported only when there is no Flexera ID available. When the Flexera ID is returned instead, this allows matching in the ARL, which then supplies its standard display name for the application instead.
(`MATCH_HOST_SW_PROD`) / `CAT_VERSION` *or* an empty string when the Flexera ID for the application is available	(`InstallerEvidence`) / `Version`	When the Flexera ID is known, the value is left empty for later supply from the ARL. For applications unknown in the ARL, the version value is imported from Data Platform.
(`MATCH_HOST_SW_PROD`) / `CAT_MANUFACTURER` *or* an empty string when the Flexera ID for the application is available	(`ImportedInstallerEvidence`) / `Publisher`	When the Flexera ID is known, the value is left empty for later supply from the ARL. For applications unknown in the ARL, the publisher's name is imported from Data Platform.

Data Platform (Table)/Column	FNMS (Table)/Column	Notes
Hard-coded values: • `FlexeraID` when it is available in (`CAT_FLEXERA_ARL`) / `ARL_ID` • `BDNA` when the Flexera ID for the application is not available.	(`ImportedInstallerEvidence`) / `Evidence`	When the Flexera ID is known, the evidence type `FlexeraID` reflects this. Otherwise, when Data Platform is unaware of any matching entry in the ARL, it is identified as the evidence type. ***Tip:*** *The internal database value BDNA is replaced with Data Platform for presentation in the web interface.*
(`CAT_FLEXERA_ARL`) / `ARL_ID` when the Flexera ID is available, *or* an empty string	(`ImportedInstallerEvidence`) / `ProductCode`	The Flexera ID (without the `arl://` prefix) is used when available. Otherwise an empty string.

Data not imported from Data Platform v5

These columns in the `ImportedInstallerEvidence` table cannot be populated by inventory imports from Data Platform v5. These missing data points do not impact license compliance calculations. For further details about these columns, please see the schema reference.

- `ComplianceConnectionID` — *internally generated*
- `AccessModeID` (relevant only to file evidence, whereas this import is of installer evidence).

Imported Computers (Inventory Devices)

The `ImportedComputer` table is a staging table used for input of inventory device records. From here, de-duplicated and normalized records are merged into the `ComplianceComputer` table.

There are two listings below. The first maps what is imported from Data Platform v5. The second lists values that FlexNet Manager Suite expects to load into the `ImportedComputer` table, that are not available from Data Platform v5, and the impact of these gaps on license reconciliation calculations where these rely on data from only this source.

Data mapping for imported inventory devices

The following listing matches the source data from Data Platform v5 with the equivalent column in the `ImportedComputer` table. Other columns from the `ImportedComputer` table cannot be populated from Data Platform v5, and so are omitted here (and shown below). For more details on the database tables and columns within FlexNet Manager Suite, please see the *FlexNet Manager Suite Schema Reference* PDF, available through the title page of online help.

Data Platform (Table)/Column	FNMS (Table)/Column	Notes
Fabricated from: (MATCH_TASK_DET) / PROCESS_ID (MATCH_TASK_DET) / TASK_NAME *and the checksum for the same* (MATCH_HOST_OS) / HOST_ID	(ImportedComputer) / ExternalID The format is of this form (shown with nonsense values): `PID123|TheTask|654321:Host789`	An external ID is fabricated from the various details shown (truncated as required and joined with various separator characters). This is then mapped through the ImportedStringMapping table to convert string IDs to integer IDs.
(MATCH_HOST_OS) / HOSTNAME	(ImportedComputer) / ComputerName	The name of the computer. • In Windows, this is the NetBIOS name of the local computer, as returned by GetComputerName() • For UNIX-like platforms, it is the host name of the machine, as returned by gethostname(2).
(MATCH_HOST_OS) / DOMAIN	(ImportedComputer) / Domain	The domain reported by the computer. ***Note:*** *Data Platform exports a domain flat name, while the ImportedComputer table handles either a flat or DNS domain name, but the DNS domain name is preferred. Watch out for possible problems merging device records obtained from multiple inventory sources.*
(MATCH_HOST_OS) / DISOVERED_OS_NAME	(ImportedComputer) / OperatingSystem	The operating system of the computer.
(MATCH_HOST_OS) / DISOVERED_OS_PATCH_LEVEL	(ImportedComputer) / ServicePack	The service pack installed for the operating system.
(MATCH_HOST_OS) / CALC_NUM_OF_PHYSICAL_CPUS	(ImportedComputer) / NumberOfProcessors	The number of processors in the computer.
(MATCH_HOST_OS) / DISC_CPU_MODEL	(ImportedComputer) / ProcessorType	The type of processor in the computer.

Data Platform (Table)/Column	FNMS (Table)/Column	Notes
(MATCH_HOST_OS) / DISC_SPEED_MHZ	(ImportedComputer) / MaxClockSpeed	The maximum clock speed of the fastest processor in the computer.
(MATCH_HOST_OS) / CALC_NUM_OF_CORES	(ImportedComputer) / NumberOfCores	The number of cores in the computer. ***Note:*** *Currently Data Platform gets an incorrect result from virtual machines (defect repair estimate 1Q2018). This affects the consumption calculations for all license types that use cores as a metric (most of these assume 1 core per processor when the number of cores is not available):* • *Device (Core-Limited)* • *Core Points* • *IBM PVU* • *Microsoft Server Core* • *Microsoft Server/ Management Core* • *Oracle Processor.*
(DISC_ALL_OS) / TOTALMEMORY	(ImportedComputer) / TotalMemory	The total RAM in the computer, in bytes.

Data Platform (Table)/Column	FNMS (Table)/Column	Notes
(MATCH_HOST_HW_PROD_MODEL) / CAT_TAXONOMY_CATEGORY2	(ImportedComputer) / ChassisType	The type of case of the computer. Some license types use this information to optimize the licensing position, particularly with desktop and laptop computers. Types used in Data Platform are mapped to types in FlexNet Manager Suite as follows: **Data Platform** / **FNMS** *Empty* / Unknown Handhelds / Tablet Notebooks / Laptop Mainframes / Main System Chassis Servers / Rack Mount Chassis *All others* / Desktop
(DISC_HDDS) / The count of entries for this device	(ImportedComputer) / NumberOfHardDrives	The number of hard drives in the computer. ***Note:*** *Collected into Data Platform only by a legacy extractor or custom extractor. FlexNet Manager Suite imports it where it is available.*

Data Platform (Table)/Column	FNMS (Table)/Column	Notes
(DISC_ALL_OS) / LOCALFILESYSTEMSPACETOTAL	(ImportedComputer) / TotalDiskSpace	The total size of all hard drives in the computer. The results differ for inventory collected from SCCM directly by FlexNet Manager Suite, or indirectly through Data Platform v5: • The FlexNet connection imports the *physical* disk space (DISK_DATA) • Data Platform imports the *logical* hard drive space (v_GS_Logical_DISK).
(DISC_NICs) / The count of entries for this device	(ImportedComputer) / NumberOfNetworkCards	The number of network cards in the computer.
(DISC_MONITORS) / The count of entries for this device	(ImportedComputer) / NumberOfDisplayAdapters	The number of graphics cards in the computer. ***Note:*** *Collected into Data Platform only by a legacy extractor or custom extractor. FlexNet Manager Suite imports it where it is available.*
(DISC_NICs) / IP_ADDRESS	(ImportedComputer) / IPAddress	The IP address of the computer.
(DISC_NICs) / MACADDRESS	(ImportedComputer) / MACAddress	The MAC address of the computer.

Data Platform (Table)/Column	FNMS (Table)/Column	Notes
(MATCH_HOST_HW_PROD_MODEL) / CAT_MANUFACTURER *Failover to* (DISC_COMPUTERSYSTEM) / MANUFACTURER	(ImportedComputer) / Manufacturer	The manufacturer of the computer hardware. Some examples include: • On Windows, the SMBios manufacturer (the WMI Manufacturer property of the 'Win32_ComputerSystem' class). • On Linux, 'Manufacturer' in the 'System Information' section resulting from the 'dmidecode' command. Sample command: 'dmidecode -s system-manufacturer' • On Solaris x86, as for Linux, with failovers first to 'sysinfo SI_HW_PROVIDER' and then to 'ModelNo'. • On Solaris SPARC, the 'sysinfo SI_HW_PROVIDER'. Typically this value is 'Sun_Microsystems' or, more recently, 'Oracle Corporation'. Failover to the 'ModelNo'. • On HP-UX, the string literal 'HP'. • On AIX, the 'modelname' system attribute preceding the comma character. For example, if the 'modelname' system attribute is 'IBM,8202-E4B', then use 'IBM'. This value is typically 'IBM'.

Data Platform (Table)/Column	FNMS (Table)/Column	Notes
(MATCH_HOST_HW_PROD_MODEL) / CAT_HARDWARE_PRODUCT *and* CAT_HW_MODEL, *joined with a space* *Failover to* (DISC_COMPUTERSYSTEM) / MODEL	(ImportedComputer) / ModelNo	The model of the computer hardware or the virtual machine. This value is defined for the context of the current execution environment, rather than the physical server that may be hosting a virtual machine or partition. For examples across operating systems, see the schema documentation.
(MATCH_HOST_OS) / SERIALNUMBER *Failover to* (DISC_ALL_OS) / SERIALNUMBER	(ImportedComputer) / SerialNo	The hardware serial number of the computer. The goal of this value is to be tied to the physical hardware, partition or virtual machine and to be as unique as possible across all computers in the organization. This is due to its use in tracking computers, particularly after an operating system rebuild. This value is also used to socialize computer inventory from different inventory sources, and is used to map virtual machine guest operating system inventory to the VM host on which the virtual machine is running.
(DISC_HOSTS_USERLOGON) / DOMAIN_USERNAME	(ImportedComputer) / LastLoggedOnUser	The DOMAIN\SAMAccountName of the user last logged onto the computer. Data Platform uses the flat domain name.
(DISC_ALL_OS) / ScanDate	(ImportedComputer) / InventoryDate	The date the computer last had inventory reported.
Hard-coded value Data Platform	(ImportedComputer) / InventoryAgent	The name of the person or tool that performed the last inventory.

Data Platform (Table)/Column	FNMS (Table)/Column	Notes
(MATCH_HOST_CPU) / DISC_NUM_OF_SOCKETS	(ImportedComputer) / NumberOfSockets	The number of sockets in the computer. ***Tip:*** *Data Platform supplies this value from Technopedia. Other inventory sources supported by FlexNet Manager Suite cannot gather this value from inventory; and in general, for compliance calculations where the number of sockets is not available in inventory, the value is approximated by using the number of processors (which is satisfactory when there are no empty sockets in the inventory device). In devices that have any empty sockets, non-blank information from Technopedia is superior. Notice that certain Oracle license types (such as Oracle Database Standard Edition) have a restriction on the number of sockets on the host server.*

Data Platform (Table)/Column	FNMS (Table)/Column	Notes
(MATCH_HOST_CPU) / CALC_NUM_OF_LOGICAL_CPUS	(ImportedComputer) / NumberOfLogicalProcessors	The number of logical processors in the computer. ***Note:*** *Currently Data Platform gets an incorrect result from virtual machines (defect repair estimate 1Q2018). This affects the consumption calculations for all license types that use processors as a metric:* • *Device (Processor-Limited)* • *Microsoft Server Processor* • *Processor* • *Processor Points.*

Data not imported from Data Platform v5

These columns in the `ImportedComputer` table cannot be populated by inventory imports from Data Platform v5. The majority of these missing data points do not impact license compliance calculations (these are listed here without comment). Those that do have an impact are noted below. For further details about these columns, please see the schema reference.

- `HardwareInventoryDate`
- `ServicesInventoryDate`
- `ComplianceConnectionID` — *internally generated*
- `ComplianceComputerID` — *internally generated*
- `ComplianceDomainID` — *internally generated*
- `IncompleteRecord` — This is always set to zero for imports from Data Platform (that is, the imported record is regarded as 'sufficient' for its normal use in license compliance calculations, subject to the known limitations described in these pages).
- `PartialNumberOfProcessors` — Absence of this value prevents subcapacity license consumption calculations for IBM PVU and Oracle Processor licenses.
- `UntrustedSerialNo`
- `FullDetailsFromExternalID`
- `FullDetailsFromComplianceConnectionID`
- `ComplianceComputerTypeID` — This is the foreign key to the `ComplianceComputerType` table, identifying

whether the device is a VM, a VM host, and so on. *Compliance impact:* Prevents (unrecognized) VDI templates consuming from a license. Prevents (unrecognized) VMs from observing license rules that require (or disallow) consumption by VMs.

- `ILMTAgentID` — For imports from ILMT, identifies the device as a candidate for subcapacity calculations on any related IBM PVU license(s). *Compliance impact:* Device consumption may be at full capacity, and calculated only when a full inventory import and compliance calculation is run (typically, once daily).
- `FNMPComputerUID`
- `HostIdentifyingNumber` — Useful for matching records from different sources, but no direct effect on compliance calculations.
- `HostType` — Useful for matching records from different sources, but no direct effect on compliance calculations.
- `IsRemoteACLDevice`
- `IsDuplicate`
- `LegacySerialNo`
- `UUID`
- `IMEI`
- `PhoneNumber` (for mobile devices)
- `EmailAddress` (for mobile devices)
- `CalculatedUser` — Typically, calculated by FlexNet Manager Suite as the most frequent user in the last ten log-on records. Data Platform v5 only reports this last logged on user name (`LastLoggedOnUser`); but this is sufficient.
- `LastSuccessfulInventoryDate`
- `MDScheduleGeneratedDate`
- `MDScheduleContainsPVUScan`
- `FirmwareSerialNumber`
- `MachineID`
- `IgnoredDueToLicense`.

Imported VMs and Hosts

The `ImportedVirtualMachine` table is a staging table used for input of virtual machine records, including their host server. From here, de-duplicated and normalized records of the VM properties are merged into the `VirtualMachine` table; and if not already present, records for both the VM and the host are added to the `ComplianceComputer` table.

Tip: *This data is in addition to the inventory imported for the virtual machine (and, separately, its host) and imported as part of the inventory device records. For details, see Imported Computers (Inventory Devices).*

There are two listings below. The first maps what is imported from Data Platform v5. The second lists values that FlexNet Manager Suite expects to load into the `ImportedVirtualMachine` table, that are not available from Data Platform v5, and the impact of these gaps on license reconciliation calculations where these rely on data from only this source.

Data mapping for imported virtualization records

The following listing matches the source data from Data Platform v5 with the equivalent column in the `ImportedVirtualMachine` table. Other columns from the `ImportedVirtualMachine` table cannot be populated from Data Platform v5, and so are omitted here (and shown below). For more details on the database tables and columns within FlexNet Manager Suite, please see the *FlexNet Manager Suite Schema Reference* PDF, available through the title page of online help.

Data Platform (Table)/Column	FNMS (Table)/Column	Notes
Fabricated from: (MATCH_TASK_DET) / PROCESS_ID (MATCH_TASK_DET) / TASK_NAME *and the checksum for the same* (DISC_VIRTUAL_HOSTGUEST) / HOST_ID	(ImportedVirtualMachine) / HostComputerID The format is of this form (shown with nonsense values): `PID123|TheTask|654321:Host789`	A computer ID for the host is fabricated from the various details shown (truncated as required and joined with various separator characters). This is then mapped through the ImportedStringMapping table to convert string IDs to integer IDs.
(MATCH_HOST_HW_PROD_MODEL) / CAT_MANUFACTURER	(ImportedVirtualMachine) / VirtualMachineType	The type of virtual machine. Values from Data Platform are transformed as follows: • Microsoft becomes VMType.HyperV • VMware becomes VMType.VMware • Any other value becomes VMType.Unknown.
(DISC_ALL_OS) / OSCOMPUTERNAME	(ImportedVirtualMachine) / ComputerName	As Data Platform has only one name for the VM, the previous value is replicated here (again).
(MATCH_HOST_HW_PROD_MODEL) / CAT_HW_MODEL *Failover to* (MATCH_HOST_HW_PROD_MODEL) / DISCOVERED_MODEL	(ImportedVirtualMachine) / ModelNo	The model number reported for the VM.
(MATCH_HOST_HW_PROD_MODEL) / CAT_MANUFACTURER	(ImportedVirtualMachine) / Manufacturer	The manufacturer visible to the guest OS running on the VM.

Data Platform (Table)/Column	FNMS (Table)/Column	Notes
Fixed value Data Platform	(`ImportedVirtualMachine`) / `InventoryAgent`	The tool that provided this inventory record.
(`DISC_ALL_OS`) / `OPERATINGSYSTEM_TYPE_LABEL`	(`ImportedVirtualMachine`) / `GuestFullName`	The operating system configured for the guest.
Fabricated from: (`MATCH_TASK_DET`) / `PROCESS_ID` (`MATCH_TASK_DET`) / `TASK_NAME` *and the checksum for the same* (`DISC_VIRTUAL_HOSTGUEST`) / `Guest_ID`	(`ImportedVirtualMachine`) / `VMComputerID` The format is of this form (shown with nonsense values): `PID123|TheTask|654321:Host789`	A computer ID for the guest is fabricated from the various details shown (truncated as required and joined with various separator characters). This is then mapped through the `ImportedStringMapping` table to convert string IDs to integer IDs.

Data not imported from Data Platform v5

The following columns in the `ImportedVirtualMachine` table cannot be populated by inventory imports from Data Platform v5. Notice that several of them (such as `NumberOfProcessors`, `NumberOfHardDrives`, and `IPAddress`) have matching data points imported as part of the inventory device import for the same guest machine. While several of these missing data points do not impact license compliance calculations, the following gaps are significant:

- When the original input to Data Platform was from BMC Discovery (ADDM), information about virtual machines (and hosts) is *not* available. In contrast, when the data source for Data Platform was Microsoft SCCM, the identification of hosts and guests *is* available for supported platforms such as Hyper-V.

  Tip: *This is the inverse of the result obtained when FlexNet Manager Suite takes inventory from these sources directly (rather than through Data Platform). In the this unmediated case, inventory from SCCM does not include the host/guest, and the inventory from ADDM does include the host/guest relationship. Of course, if you have multiple inventory sources collecting from a common target device, the results are merged within FlexNet Manager Suite to provide the most complete record possible.*

- When data is available (from inventory sources such as Microsoft SCCM into Data Platform), the import from Data Platform to FlexNet Manager Suite is missing information about resource pools, vMotion, and partitions. This means that, when Data Platform v5 is the only inventory source for these machines, compliance calculations are impossible for consumption from some licenses such as IBM PVU, Microsoft, or Oracle server licenses.

For further details about these columns that are missing data, please see the schema reference.

- `VMName`
- `VCObjectID`
- `FriendlyName`
- `UUID`
- `TotalMemory`

- PoolName
- CPUUsage
- MemoryUsage
- MaxNumberOfLogicalProcessors
- VMEnabledStateID
- NumberOfProcessors
- ProcessorType
- NumberOfNetworkCards
- ComplianceConnectionID
- VMLocation
- PoolType
- ZoneResourceManagementMethodType
- AffinityEnabled
- CPUAffinity
- CoreAffinity
- PartitionID
- PartitionNumber
- FullComputerName
- IPAddress.

Imported Installation Records

The ImportedInstalledInstallerEvidence table is a staging table used to collect records of which installation evidence has been found on what inventory devices. From here, de-duplicated and normalized records are merged into the InstalledInstallerEvidence table.

InstalledInstallerEvidence is a simple table that mainly links installer evidence records with inventory device records. In the case of imports from Data Platform v5, only the two external IDs are needed, with other values supplied by FlexNet Manager Suite as and when required. The first listing below maps the two columns from Data Platform source to FlexNet Manager Suite destination. Below is a list of the remaining columns in the ImportedInstalledInstallerEvidence table that are not populated by this import (and do not need to be).

Data mapping for imported installed installer evidence

The following listing matches the source data from Data Platform v5 with the equivalent column in the

ImportedInstalledInstallerEvidence table. For more details on the database tables and columns within FlexNet Manager Suite, please see the *FlexNet Manager Suite Schema Reference* PDF, available through the title page of online help.

Data Platform (Table)/Column	FNMS (Table)/Column	Notes
(MATCH_HOST_SW_PROD) / CAT_SW_UUID	(ImportedInstalledInstallerEvidence) / ExternalInstallerEvidenceID	The same value as the (ImportedInstallerEvidence) / ExternalInstallerID.
Fabricated from: (MATCH_TASK_DET) / PROCESS_ID (MATCH_TASK_DET) / TASK_NAME *and the checksum for the same* (MATCH_HOST_SW_PROD) / HOST_ID	(ImportedInstalledInstallerEvidence) / ExternalID The format is of this form (shown with nonsense values): `PID123|TheTask|654321:Host789`	An external ID for the inventory device is fabricated from the various details shown (truncated as required and joined with various separator characters). This is then mapped through the ImportedStringMapping table to convert string IDs to integer IDs. The same value as the (ImportedComputer) / ExternalID.
(MATCH_HOST_SW_PROD) / DISCOVERED_INSTALLDATE	(ImportedInstalledInstallerEvidence) / InstallDate	The installation date recorded for the installer evidence.

Data not imported from Data Platform v5

These columns in the ImportedInstalledInstallerEvidence table cannot be populated by inventory imports from Data Platform v5. These missing data points do not impact license compliance calculations. For further details about these columns, please see the schema reference.

- ComplianceConnectionID — *internally generated*
- ExternalInstanceID
- DiscoveryDate (will be set to the date of first import from this connection).

Imported Users

The ImportedUser table is a staging table used for input of records about users. From here, de-duplicated and normalized records are merged into the ComplianceUser table.

ComplianceUser stores information about end-users in the enterprise, including contact details, login details, and inventory source details (if applicable). End-users are the people using computers within your enterprise (as distinct from operators, who are people permitted to use FlexNet Manager Suite).

There are two listings below. The first maps what is imported from Data Platform v5. The second lists values that FlexNet Manager Suite expects to load into the ImportedUser table, that are not available from Data Platform v5.

Data mapping for imported users

The following listing matches the source data from Data Platform v5 with the equivalent column in the `ImportedUser` table. Other columns from the `ImportedUser` table cannot be populated from Data Platform v5, and so are omitted here (and shown below). For more details on the database tables and columns within FlexNet Manager Suite, please see the *FlexNet Manager Suite Schema Reference* PDF, available through the title page of online help.

Data Platform (Table)/Column	FNMS (Table)/Column	Notes
(`DISC_HOSTS_USERLOGON`) / `USERDOMAIN` and `USERLOGON` combined	(`ImportedUser`) / `ExternalID`	Mapped through the `ImportedStringMapping` table to convert string IDs to integer IDs.
(`DISC_HOSTS_USERLOGON`) / `USERDOMAIN`	(`ImportedUser`) / `Domain`	The domain of the end-user. Data Platform exports the flat domain name.
(`DISC_HOSTS_USERLOGON`) / `USERLOGON`	(`ImportedUser`) / `SAMAccountName`	The login name (SAM account name) of the end-user.
Hard-coded value `Data Platform`	(`ImportedUser`) / `InventoryAgent`	The name of the person or tool that performed the last inventory.

Data not imported from Data Platform v5

These columns in the `ImportedUser` table cannot be populated by inventory imports from Data Platform v5. These missing data points do not impact license compliance calculations. For further details about these columns, please see the schema reference.

- `ComplianceConnectionID` — *internally generated*
- `UserName`
- `ComplianceDomainID`
- `LocationID`
- `FirstName`
- `LastName`
- `Email`
- `EmployeeNumber`
- `CostCenter` (as reported in SAP)
- `ComplianceUserID` — *internally generated*
- `ComplianceDomainID` — *internally generated*
- `IsBlacklisted` — *internally generated*
- `MapUsingEmailAddress`.

Imported Software Usage

The `ImportedInstalledInstallerEvidenceUsage` table is a staging table used to collect records of actual use of applications related to installation evidence found on inventory devices. From here, de-duplicated and normalized records are merged into the `InstalledSoftwareUsageData` table.

There are two listings below. The first maps what is imported from Data Platform v5. The second lists values that FlexNet Manager Suite expects to load into the `ImportedInstalledInstallerEvidenceUsage` table, that are not available from Data Platform v5, and the impact of these gaps on license reconciliation calculations where these rely on data from only this source.

Data mapping for imported usage related to installer evidence

The following listing matches the source data from Data Platform v5 with the equivalent column in the `ImportedInstalledInstallerEvidenceUsage` table. Other columns from the `ImportedInstalledInstallerEvidenceUsage` table cannot be populated from Data Platform v5, and so are omitted here (and shown below). For more details on the database tables and columns within FlexNet Manager Suite, please see the *FlexNet Manager Suite Schema Reference* PDF, available through the title page of online help.

Data Platform (Table)/Column	FNMS (Table)/Column	Notes
(`MATCH_SOFTWARE_METERING`) / `USAGE_YEAR` *and* `USAGE_MONTH_N` Failover to (`MATCH_SW_RECENTLY_USED_APPS`) / `LASTUSEDDATE`	(`ImportedInstalledInstallerEvidenceUsage`) / `StartDate`	The start date is set to a string formatted similarly to ISO 8601 (except for the slash separator: 2019/04/01) by conjoining the year and month-number from Data Platform, and adding an assumed day for the first of the month. Only if there is no value available by this path, we use the date available from the alternate table shown.
Fabricated from: (`MATCH_TASK_DET`) / `PROCESS_ID` (`MATCH_TASK_DET`) / `TASK_NAME` *and the checksum for the same* (`MATCH_SOFTWARE_METERING`) / `HOST_ID`	(`ImportedInstalledInstallerEvidenceUsage`) / `ExternalID` The format is of this form (shown with nonsense values): `PID123|TheTask|654321:Host789`	An external ID is fabricated from the various details shown (truncated as required and joined with various separator characters). This is then mapped through the `ImportedStringMapping` table to convert string IDs to integer IDs. ***Tip:*** *If the `MATCH_SOFTWARE_METERING` table cannot provide a host ID, it is taken from `MATCH_SW_RECENTLY_USED_APPS`.*

Data Platform (Table)/Column	FNMS (Table)/Column	Notes
(MATCH_SOFTWARE_METERING) / CAT_SW_UUID	(ImportedInstalledInstaller EvidenceUsage) / ExternalInstallerID	The identifier used in Data Platform for the installer evidence. ***Tip:*** *If the MATCH_SOFTWARE_METERING table cannot provide an external ID, it is taken from MATCH_SW_RECENTLY_USED_APPS.*
(MATCH_SOFTWARE_METERING) / USAGECOUNT *plus* TSUSAGECOUNT	(ImportedInstalledInstaller EvidenceUsage) / NumberOfSessions	If the number of sessions in which the evidence was in use cannot be calculated by this method, the default value 1 is inserted.
(MATCH_SOFTWARE_METERING) / LASTUSEDDATE Failover to (MATCH_SW_RECENTLY_USED_APPS) / LASTUSEDDATE	(ImportedInstalledInstaller EvidenceUsage) / LastUsedDate	The last known date when usage of the application is recorded (in one table or the other).
(MATCH_SW_RECENTLY_USED_APPS) / LASTUSERDOMAIN, *then a backslash* \, *then* LASTUSERLOGON	(ImportedInstalledInstaller EvidenceUsage) / ExternalUserID	The end-user identified in the source as using the software. The identifier is fabricated by conjoining the flat domain name and logon name (separated by a backslash).

Data not imported from Data Platform v5

These columns in the ImportedInstalledInstallerEvidenceUsage table cannot be populated by inventory imports from Data Platform v5. These missing data points do not impact license compliance calculations. For further details about these columns, please see the schema reference.

- ComplianceConnectionID — *internally generated*
- ExternalInstanceID (used only for Oracle).

VII

HP DDMI Adapter

This part introduces the FlexNet adapter for importing data from HP DDMI (for those enterprises that have not yet migrated to HPE Universal Discovery, for which see HPE Universal Discovery Adapter).

1

Purpose and Architecture of the HP DDMI Adapter

The FlexNet adapter for HP Discovery and Dependency Mapping Inventory (HP DDMI) collects data from your HP DDMI implementation, and stores it in an intermediate staging database. From there, the FlexNet Manager Suite importers bring the data into the compliance database so that you have a single, unified view of all inventory sources, and of the impact that the collected inventory has on your license position.

The work is done in two separate stages, and by two separate components:

- The FlexNet DDMI *agent* reads inventory directly from the HP DDMI incoming inventory (`.xsf`) files, and writes the data into the staging database.
- The FlexNet DDMI *adapter connection* connects to the staging database, massages the data into the format required for integration with FlexNet Manager Suite, and imports it to the compliance database.

This architecture is summarized in the following diagram, with the numbers described below:

1. XSF files (compressed XML files) are produced by HP agents and uploaded to a central HP DDMI server.
2. XSF files are imported into the databases for HP DDMI and HP uCMDB. These two stages are the normal operation of HP DDMI, and are unaffected by the presence of the FlexNet adapter.
3. The first component of the FlexNet adapter for HP DDMI (an agent which may be installed on the HP DDMI server, where the XSF files are uploaded, or on any other device with read access to the same collection of XSF files) reads the uploaded files and writes the data into the staging database required for this adapter. Copying the data from the original XSF files allows FlexNet Manager Suite to capture evidence that, if unrecognized by HP DDMI, is otherwise discarded.
4. The data import connection (configured from an inventory beacon, which is not shown in this simplified diagram) links to the staging database, extracts the latest data, and uploads it to the compliance database for FlexNet Manager Suite.

2

Installation and Configuration

Everything needed to set up the FlexNet adapter for HP DDMI is included in the Adapter Tools archive available from the Flexera Customer Community. You need credentials supplied by Flexera to access this download. The download includes:

- Two SQL scripts to define the schema for, and populate stored procedures for, the staging database.
- The agent executable `FnmpDDMIAgent.exe`, and an accompanying `CommandLine.dll` to enable command-line use.

This chapter includes all the details needed for installation and configuration. There are validation notes are included at each stage of the set-up.

This chapter assumes that you have a functional implementation of HP DDMI, and an account with sufficient privileges on your HP DDMI server to install the agent executable there (or similar in your chosen location, if you are placing the agent elsewhere).

Download Adapter Tools Archive

The Adapter Tools archive includes content for many adapters, and is updated on the Flexera website from time to time. Start this procedure using a web browser on a computer that has good network accessibility from all the machines needing installations for your adapter.

To download the adapter tools archive:

1. Download the `Adapter Tools for` FlexNet Manager Suite `2019 R2.zip` archive from the Flexera Customer Community knowledge base:

   a. Access https://flexeracommunity.force.com/customer/articles/en_US/INFO/Adapter-Tools-for-FlexNet-Manager-Suite.

   Tip: *Access requires your Customer Community user name and password. If you do not have one, use the link on the login page to request one.*

   b. Click the link `Adapter Tools for FlexNet Manager Suite`.

   A new browser tab may appear temporarily, and the download of `Adapter Tools for` FlexNet Manager

Suite 2019 R2.zip commences.

c. In your browser dialog, choose to save the file, and if the browser allows it, direct the saved file to a convenient working location (such as C:\temp on a central, accessible server).

If your browser saves the file to a default location (such as your Downloads folder), move or copy it to the appropriate working location when the download is finished.

2. Right-click the zip archive, and choose **Extract All...**.

The folders are now available for the range of adapters in the Adapter Tools archive.

Creating the Staging Database

The FlexNet adapter for HP DDMI requires a staging database, which is used as follows:

- The agent installed on your HP DDMI server reads the DDMI inventory files directly, performs minimal data rationalization, and saves the results to the staging database
- The connection from FlexNet Manager Suite reads the data from the staging database, shapes it appropriately, and uploads it to the compliance database.

The staging database can be created on any convenient instance of Microsoft SQL Server that meets the following criteria:

- Provides high-speed network access from the HP DDMI server (or other location of the installed agent), and from an appropriate inventory beacon that can upload to the application server for FlexNet Manager Suite
- Allows read/write access to the account running the FlexNet agent on the HP DDMI server, and read access by the service account running the BeaconEngine service on the inventory beacon (or you may choose to implement either a special Windows account and an SQL account to collect the DDMI staged data)
- Is running an instance of Microsoft SQL Server 2012 or later.

The staging database size will be roughly equivalent to the sum of the inventory (.xsf) files uploaded to your HP DDMI server. The agent overwrites any previous record from the same device with the latest inventory upload, to that the net database size does not grow appreciably larger than the sum of the source inventory files.

Your download of the Tier 1 adapters archive (see Download Adapter Tools Archive) included two SQL scripts that can assist with creating the staging database. You can achieve this through SQL Server Management Studio, or from the command line as described in the following procedure. Use whichever approach best suits your enterprise practices.

To create the staging database from the command line:

1. Navigate through the unzipped tools archive to Tier 1 Adapter Tools > HP Discovery and Dependency Mapping Inventory Tools > SQL.
2. If necessary, copy the script DDMIStagingSchema.sql from the this folder of your unzipped adapter archive to a temporary folder on your staging database server.
3. Logged in on the database server as an account with read/write access to the database, open a command prompt.
4. In the command prompt window, execute the following command, as amended:

```
sqlcmd -S ServerName\InstanceName -i TemporaryPath\DDMIStagingSchema.sql
```

where:

- *ServerName* is the name of the server hosting Microsoft SQL Server 2012 or later. Use a "`.`" (dot) if you are running the staging script on the same server as the database instance; or if you are adapting these instructions to a different environment, insert the server name or its IP address.
- *InstanceName* is the name of the database instance to use for the database staging tables. You can omit this parameter (and the backslash separator) if the default instance name is being used.
- *TemporaryPath* is the location where you saved the SQL procedure.

Example:

```
sqlcmd -S .\Development -i C:\temp\DDMIStagingSchema.sql
```

The database `DDMIStaging` is created with all necessary tables, indices, and so on.

5. Similarly, execute the second script (with the same parameter substitution) to create the stored procedures necessary in the staging database.

```
sqlcmd -S ServerName\InstanceName -i TemporaryPath\DDMIStagingProcedures.sql
```

Example:

```
sqlcmd -S .\Development -i C:\temp\DDMIStagingProcedures.sql
```

The staging database is now ready for operation.

Tip: *If you wish to do so, you may now rename this database. Keep the name handy for the configuration processes that follow.*

The schema for the staging database (below) is, in the main, self-explanatory, reflecting the content of the `.xsf` files found on your HP DDMI server. The one unique table is `InventoryImportDate`. This table lists the names of inventory files that have already been loaded into the staging database, and saves the last modified date of each file as its most recent inventory date. When the FlexNet agent scans through the inventories for possible import into the staging database, it uses this table to manage differential imports, only importing an inventory file if it is newer than the last recorded inventory file of that name that was imported.

StagedComputer

Column Name	Data Type	Allow Nulls
ExternalComputerID	int	☐
AssetTag	nvarchar(256)	☐
ComputerName	nvarchar(256)	☑
DomainQualifiedName	nvarchar(100)	☑
DomainFlatName	nvarchar(32)	☑
OperatingSystem	nvarchar(128)	☑
ServicePack	nvarchar(128)	☑
NumberOfProcessors	int	☑
ProcessorType	nvarchar(256)	☑
MaxClockSpeed	int	☑
NumberOfCores	int	☑
NumberOfSockets	int	☑
TotalMemory	bigint	☑
ChassisType	nvarchar(128)	☑
ChassisSerialNumber	nvarchar(100)	☑
NumberOfHardDrives	int	☑
TotalDiskSpace	bigint	☑
NumberOfNetworkCards	int	☑
NumberOfDisplayAda...	int	☑
IPAddress	nvarchar(256)	☑
MACAddress	nvarchar(256)	☑
Manufacturer	nvarchar(128)	☑
ModelNo	nvarchar(128)	☑
SerialNo	nvarchar(100)	☑
LastLoggedOnUser	nvarchar(64)	☑
InventoryDate	datetime	☑
InventoryAgent	nvarchar(128)	☑
		☐

StagedDomain

Column Name	Data Type	Allow Nulls
QualifiedName	nvarchar(100)	☐
Flatname	nvarchar(32)	☑
		☐

StagedInstallerEvidence

Column Name	Data Type	Allow Nulls
ExternalComputerID	int	☐
ExternalInstallerID	int	☐
DisplayName	nvarchar(256)	☐
Version	nvarchar(72)	☑
Publisher	nvarchar(200)	☑
Evidence	nvarchar(32)	☑
		☐

StagedFileEvidence

Column Name	Data Type	Allow Nulls
StagedFileEvidenceID	int	☐
FileName	nvarchar(256)	☐
FileVersion	nvarchar(100)	☑
ProductVersion	nvarchar(100)	☑
ProductName	nvarchar(100)	☑
Company	nvarchar(100)	☑
Description	nvarchar(100)	☑
FileSize	int	☑
Language	nvarchar(200)	☑
		☐

StagedComputerFileEvidence

Column Name	Data Type	Allow Nulls
ExternalFileID	int	☐
ExternalComputerID	int	☐
StagedFileEvidenceID	int	☐
FilePath	nvarchar(400)	☑
		☐

StagedUser

Column Name	Data Type	Allow Nulls
ExternalUserID	int	☐
UserName	nvarchar(64)	☑
Domain	nvarchar(100)	☐
SAMAccountName	nvarchar(64)	☐
		☐

StagedWMIEvidence

Column Name	Data Type	Allow Nulls
ExternalWMIEvidenceID	int	☐
ExternalComputerID	int	☐
ClassName	nvarchar(50)	☐
PropertyName	nvarchar(50)	☐
PropertyValue	nvarchar(256)	☐
		☐

InventoryImportDate *

Column Name	Data Type	Allow Nulls
InventoryFileName	nvarchar(256)	☐
InventoryDate	datetime	☑
		☐

Configuring the FlexNet Agent for HP DDMI

The FlexNet agent for HP DDMI is the code entity that accesses the inventory (`.xsf`) files uploaded to your HP DDMI server, and saves the resulting data to the staging database that you have just established. Operationally, it consists of two parts:

- A small C# application (an `.exe` and a `.dll`) that acts only as a transport, decompressing the `.xsf` files (which are compressed XML) and feeding them to the second part
- A stored procedure, already created in the staging database (see Creating the Staging Database), that unpacks the XML elements and saves the resulting data in the staging database.

Clearly, it is only the first of these that is the focus of this topic.

The agent may be installed anywhere, as long as it has access to the folder on your HP DDMI server where the `.xsf` inventory files are uploaded, and to the staging database. Optional locations include:

- On the HP DDMI server (optimal from a network performance viewpoint)
- On the server hosting the staging database
- On an inventory beacon that has fast network access to both of the above
- On any other computer meets that same requirement of fast network access to the data source and destination.

Although much of the inventory processing load is delegated to the SQL stored procedure, you can further control the impact of the executable on its host device by specifying the number of CPU threads it devotes to throughput of the `.xsf` files. In any case, the agent is normally scheduled to run only once daily in off-peak hours.

To configure the agent executable:

1. Navigate through the unzipped tools archive to `Tier 1 Adapter Tools` > `HP Discovery and Dependency Mapping Inventory Tools` > `DDMIAgent`.
2. Copy both the `FnmpDDMIAgent.exe` and `CommandLine.dll` files to their new operational location.

   Important: *Both files must be installed in the same folder.*

3. Identify or configure the account that is to run the agent.

   A suitable account:

   - Has execute rights on the device where the agent is installed
   - Has at least read-only privileges on the folder where the `.xsf` inventory files are saved on your HP DDMI server
   - Has read/write privileges on the staging database.

   You may prefer to make this a service account to prevent interactive use.

4. Configure a Windows schedule task (or use your preferred scheduling tool) to trigger execution of the agent running as your selected account on your preferred schedule.

   Choose a schedule to suit your environment. This should allow enough time to:

- Process all new `.xsf` files into the staging database
- Upload the results to the compliance database for FlexNet Manager Suite
- Align with the nightly full import and license compliance calculations, by default scheduled for 2am nightly.

A suggested schedule is daily at 10pm.

The task must invoke the `FnmpDDMIAgent.exe` executable with a command line that includes some of the following parameters (the agent also supports `-h` to list command-line options):

```
FnmpDDMIAgent.exe
         -s PathToInventories
         -d StagingDBConnectionString
         -l LoggingLevel
         -t TimeOutSeconds
         -n ThreadCount
```

where the parameters are:

Switch	M/O	Value
`-s`	Mandatory	The location of the HP DDMI inventory files (no default value). The path may be enclosed in double quotation marks, which are mandatory when there is any white space in the file path. Example: `"C:\HPDDMI\Inventories"`
`-d`	Mandatory	The connection string for the staging database (no default value). Example: `"server=172.16.38.83;Database=DDMIStaging;Trusted_Connection=yes"`
`-l`	Optional	An integer in the range 0-3 for the level of logging: • 0: no logging (default value) • 1: Errors only • 2: Errors and warnings • 3. Verbose logging.
`-t`	Optional	The number of seconds the executable should wait for a response to any command given to the staging database. Default value is `3600` (an hour).
`-n`	Optional	An integer in the range 1-20 (default 5), being the number of CPU threads the agent should use for processing the `.xsf` inventory files. Each thread invokes the database stored procedures independently. ***Tip:*** *Depending on the hosting device architecture, optimum throughput is achieved somewhere within the range of 1-20 threads, after which increasing the number of threads overloads the system and decreases performance. Given the variety of factors involved, there is no real substitute for experiment to determine the optimum value for your environment.*

Example usage (all on one line):

```
FnmpDDMIAgent.exe
        -s "C:\HPDDMI\Inventories"
        -d "server=172.16.38.83;Database=DDMIStaging;Trusted_Connection=yes"
        -l 2
```

When thus invoked, the agent reads through all updated XSF files in the source directory, and uploads their data into the staging database. Files unchanged in the nominated path since the last upload are ignored.

5. On the assumption that you already have `.xsf` files in the nominated folder on your HP DDMI server, you may wish to run the agent once now, and use Microsoft SQL Server Studio to validate the content is stored in the staging database. (Loading the staging database also provides material for testing the next stage of the process.) If you enabled logging, you may also check the log file, which is written to the same folder where the executable is running.

Configuring Upload and Import Connection

With data arriving in the staging database, it's time to configure its upload to the central application server for FlexNet Manager Suite, followed by its import to the compliance database. This process is configured on, and managed by, an inventory beacon.

To configure the connection for upload from the staging database:

1. Run the inventory beacon interface (for example, in the Windows Start menu, search for **FlexNet Beacon**, right-click it, and select **Run as administrator**).

   Tip: *Remember that you must run the inventory beacon software with administrator privileges.*

2. Open the **Inventory systems** page.
3. Expand the pull-down beside the **New...** button below the list, and choose `SQL Server`.

   The **Create SQL Source Connection** dialog appears.

4. Complete the details to define the connection to the staging database:

   a. Create a distinctive **Connection Name** that is easily recognized in the list of connections (even when the columns are narrow).

      For example: `HPDDMI Staging`.

   b. For **Source Type**, you must choose `HP Discovery and Dependency Mapping Inventory (DDMI)`.

   c. Provide the host name or IP address (in IPv4 format) for the **Server** hosting the staging database.

   d. Choose the **Authentication** method that the `BeaconEngine` process should use to connect to the database:

      - If you select `Windows Authentication`, ensure that the service account that runs the

`BeaconEngine` process on this inventory beacon has (as a minimum) read access to the staging database.

- With either of the remaining two options, specify the **Username** and **Password** in the next two fields. Of course, validate that the nominated account has (as a minimum) read access to the staging database.

e. Use the **Test connection** button to validate that you have specified the details correctly, and adjust if necessary.

f. For the **Overlapping Inventory Filter**, accept the default **Import the inventory from this source for possible merging** selection (unless you are in the process of migrating inventory collection away from HP DDMI, in which case see the online help for the **Inventory systems** page).

g. Click **Save** to close this dialog and add your new connection to the list in the **Inventory systems** page.

5. If you do not already have a suitable schedule for this connection available on this inventory beacon, select the **Scheduling** page, and click **New...** to open the **Edit Schedule** dialog and define one.
6. Back on the **Inventory systems** page, select the new connection in the list, and click **Schedule...** to select your appropriate schedule in the **Select schedule** dialog. Click **OK** to confirm your selection.
7. In the main **FlexNet Beacon** interface, click **Save** to store your newly scheduled connection.
8. If you have already run the agent to populate your staging database, you may test the new connection by keeping it selected in the list, and clicking **Execute Now**.

   The upload, the resulting import (which follows immediately), and the subsequent license compliance calculation may take some time to complete. You may monitor progress in the web interface for FlexNet Manager Suite — navigate to the system menu (⚙ ▼ in the top right corner) and choose **System Health > System Tasks**, and see the online help for that page. When the process is complete, you can inspect the inventory imported from HP DDMI in the **All Inventory** page.

Tip: *In the **All Inventory** page:*

a. *Add the **Connection name** to the listing from the column chooser*

b. *Display the simple filter bar*

c. *Filter for the name you have your HP DDMI connection (the suggestion was `HPDDMI Staging`).*

VIII

HPE Universal Discovery Adapter

You can use the HPE Universal Discovery (HPE-UD) adapter tool to collect and import inventory data from HPE Universal Discovery System to FlexNet Manager Suite. The HPE-UD adapter fetches all hardware, software, and virtualization information from the HPE-UD system and stores it in the compliance database maintained by FlexNet Manager Suite. The HPE-UD adapter support is available only with FlexNet Manager Suite version 2015 and later.

Note: *The HPE-UD adapter tool works only with version 10.10, 10.11, 10.33–11.5 of the HPE Universal Discovery System. These versions do not support Solaris 11 zones, for which reason the HPE-UD adapter cannot import these zones. Since host serial number and zone name are used for rationalizing duplicate inventory records across different inventory sources, this may mean that a device imported through the HPE-UD adapter cannot be merged with another record of the same device imported through another inventory source that supports zones information.*

1

Selecting a Configuration

This chapter gives an overview of the architecture, working, and configuration of the HPE-UD adapter. Choose the appropriate configuration for your enterprise before you implement the adapter.

Architecture and Working of the HPE Universal Discovery Adapter

The HPE-UD adapter architecture has the following main components:

- **HPE-UD Server:** This server has the `HPUD uCMDB` database and HPE discovery tools installed on it. The `HPUD uCMDB` database stores the HPE-UD inventory details.
- **Staging Server:** This server contains the Flexera `HPuCMDBStage.exe` tool and a staging database. The `HPuCMDBStage.exe` tool retrieves topology maps by executing TQL queries on `HPUD uCMDB`. The retrieved topology maps are stored in a staging database (`uCMDB_Staging`) present on the staging server. The query execution is performed using the web service interface of `HPUD uCMDB`.
- **FlexNet Manager Suite:** The Compliance Reader component residing on this server fetches data from the staging database and loads it into the FlexNet Manager Suite database when you run an inventory import.

The following diagram shows the architecture and working of the HPE-UD adapter:

Figure 5: Architecture and working of the HPE-UD adapter

1. The `HPuCMDBStage.exe` tool extracts the inventory information through the web API of `HPUD uCMDB`. This tool executes Topology Query Language (TQL) queries on the `HPUD uCMDB` database and retrieves responses in the form of topology maps. With the default configuration, this tool stores the retrieved topology maps directly into the staging database without storing them in XML files. However, you can configure this tool to write the retrieved

topology maps to XML files and then write data from those XML files to the staging database.

Note: *One topology map is generated for each TQL query executed on HPUD uCMDB.*

2. The Compliance Reader component of FlexNet Manager Suite collects data from the staging database. When you run an inventory import on the FlexNet Manager Suite server, the data extracted from the staging database is written to the compliance database on the application server

The Adapter Executable

HPE Universal Discovery (HPE-UD) adapter uses the `ucmdb-api` to obtain the version of the `uCMDB` instance. It uses the `UcmdbService` web service API to retrieve the topology maps by executing TQL queries on the `uCMDB` instance. You can retrieve the WSDL definition of the `UcmdbService` web service API from `http://<uCMDB Server>/axis2/services/UcmdbService?wsdl`. The configuration of the staging tool includes the TQL queries that will be executed on the `HPUD uCMDB` database. You don't have to import the TQL queries into the `HPUD uCMDB` database and there is no need to deploy and maintain the TQL queries on every HPE-UD instance of your enterprise.

Note: *The URL to retrieve the WSDL definition of the `UcmdbService` web service API may use 'https' instead of 'http'. It may also include a port number. Please check with your HPE-UD system administrator for more details on the web service URL.*

The tool to query the web API has the following two parts:

- `HPuCMDBStage.exe` — A console program capable of querying the `UcmdbService` API of HPE-UD and writing the results into an SQL Server database, and optionally to XML files on the local file system. This program supports command line arguments, available using `HPuCMDBStage.exe -h`. Here is the list of available command line options.

```
-h                            This help
-x <settings file>           Settings file
-s <address or URL>          uCMDB server address or URL
-u <user name>               uCMDB export user name
-p <password>                uCMDB export user password
-c <connection string>       SQL Server staging database connection string
-m <staging method>          Set staging method (stage/staged/prestaged/stream)
-f <staging file path>       Set the staging file path
```

- `FNMPuCMDBSettings.xml` — The self-documenting configuration file for `HPuCMDBStage.exe` which contains TQL queries executed against HPE-UD, and can include connection settings for HPE-UD and SQL Server.

In operation, the executable, `HPuCMDBStage.exe`, extracts the inventory data from HPE-UD and saves it for further processing. The value of the `method` parameter determines how the data is saved. The `method` parameter can have the following values:

- `Stage` — This option enables you to save inventory data from HPE-UD to a series of XML files on the staging server **(HPE-UD to XML)**. The XML files are not imported into the staging database, but you can review the extracted inventory data. This option also enables you to collect inventory data from the HPE-UD servers that are not directly connected to the staging server. You can manually copy and upload the inventory XML files to the staging server. Also

see the `Prestaged` method below.

- `Staged` — This options enables you to write the data extracted from the HPE-UD servers to XML files and then copy the information to the staging database **(HPE-UD to XML/SQL)** where it can be imported into FlexNet Manager Suite for use in compliance calculations.
- `Prestaged` — This option takes information stored in XML files (from the Stage and Staged method) and loads it into a staging SQL database **(XML to SQL)**. Inventory is not gathered from HPE-UD in this mode.
- `Stream` — This option enables you to extract the inventory information from HPE-UD and load it into the staging database directly, without storing it in XML files on the staging server **(HPE-UD to SQL)**. You can import these files to into FlexNet Manager Suite for use in compliance calculations.

You can set the default values for the `method` parameter and all other parameters in the `FNMPuCMDBSettings.xml` file. The adapter tool uses the default parameter values when you use it without other command-line options. The settings file is self-documented and the matching command-line options are available using `HPuCMDBStage.exe -h`.

Files Created by the Staging Tool

The following files are created when the staging tool performs a read/write operation on the local disk of the staging server. You can view the contents of the following files to review HPE-UD configuration item details that have been extracted.

Filename	Content
`Computer.xml`	Details of computers and their properties.
`Virtualization.xml`	Virtualization and partitioning details.
`InstalledSoftware.xml`	Installed software evidence linked to each computer.

2

Installation and Configuration

For **on-premises** implementations, the adapter XML files are included as a part of the FlexNet Manager Suite installation. You need to download the staging tool, the configuration file, and the staging schema for this tool from `https://flexerasoftware.flexnetoperations.com`, or from the Flexera Customer Community.

You need credentials supplied by Flexera to access this download. Details of the download are included in Download Adapter Tools Archive.

- The HPE-UD adapter suits FlexNet Manager Suite releases 2015 and later for on-premises delivery.
- The build number for this adapter is 10.3.0.12006 (or higher). You can identify this number by right-clicking the `HPuCMDBStage.exe` file in Windows Explorer, selecting **Properties**, and looking at the **Details** tab.

Save the zipped archive to a suitable temporary location, and unzip it.

Full details of setting up the HPE-UD adapter are included in this chapter. The following chapter (see HPE-UD Adapter Operation) covers testing the completed installation and validating the results.

Download Adapter Tools Archive

The Adapter Tools archive includes content for many adapters, and is updated on the Flexera website from time to time.

Start this procedure using a web browser on a computer that has good network accessibility from all the machines needing installations for your adapter.

To download the adapter tools archive:

1. Download the `Adapter Tools for` FlexNet Manager Suite `2019 R2.zip` archive from the Flexera Customer Community knowledge base:
   a. Access https://flexeracommunity.force.com/customer/articles/en_US/INFO/Adapter-Tools-for-FlexNet-Manager-Suite.

   Tip: *Access requires your Customer Community user name and password. If you do not have one, use the link on the login page to request one.*

b. Click the link `Adapter Tools for FlexNet Manager Suite`.

   A new browser tab may appear temporarily, and the download of `Adapter Tools for` FlexNet Manager Suite `2019` R2`.zip` commences.

c. In your browser dialog, choose to save the file, and if the browser allows it, direct the saved file to a convenient working location (such as `C:\temp` on a central, accessible server).

   If your browser saves the file to a default location (such as your `Downloads` folder), move or copy it to the appropriate working location when the download is finished.

2. Right-click the zip archive, and choose **Extract All...**.

The folders are now available for the range of adapters in the Adapter Tools archive.

Selecting a Staging Server

Multiple configuration options are there to configure the staging server. You can install the staging server on any of the following types of computers:

- A dedicated stand-alone server or a virtual machine.
- Any other suitable machine in your enterprise, such as a print server.
- An inventory beacon.
- The central application server where FlexNet Manager Suite is installed.

The following are the requirements for a staging server:

- Microsoft Windows 2008 or later.
- Access to an instance of Microsoft SQL Server 2012 or later to host the staging database. You can implement the staging database on the staging server or on a separate database server.
- Microsoft .NET 4.0 run time environment installed.
- Efficient network access to each HPE-UD server in your enterprise, using the HTTP or HTTPS protocols.

Creating the Staging Database Tables

Once your selected staging server is able to access the Microsoft SQL Server instance, you can use the provided script to create the staging database and set up the appropriate database tables. You can achieve this through SQL Server Management Studio, or from the command line as described in the following procedure.

To create the staging database from the command line:

1. Download and install the Adapter Tools archive. For more information, see Download Adapter Tools Archive.
2. Navigate through the unzipped archive to `Adapter Tools` > `HP Universal Discovery Tools` > `SQL`.
3. If necessary, copy the script `uCMDB_staging.sql` from the `SQL\` folder of your unzipped adapter archive to a temporary folder on your staging server.

4. Open a command prompt on the staging server.
5. In the command prompt window, execute the following command, as amended:

```
sqlcmd -S ServerName\InstanceName -i TemporaryPath\uCMDB_staging.sql
```

where:

- The database `uCMDB_staging` is created with all necessary tables, indices, and so on.
- *`ServerName`* is the name of the server hosting Microsoft SQL Server 2012 or later. You can install the database on the staging server or on any remote server. You can use the name of the staging server or its IP address, or a "`.`" (dot) if you are running the staging script on the same server as the database instance.
- *`InstanceName`* is the name of the database instance to use for the database staging tables. You can omit this parameter if the default instance name is being used.
- *`TemporaryPath`* is the location where you saved the SQL procedure.

Example:

```
sqlcmd -S 192.100.0.20\Development -i C:\temp\uCMDB_staging.sql
```

The staging database is now ready for operation.

Configuring HPE Universal Discovery System

You must configure the HPE-UD adapter settings to include a user which can perform the Run Legacy API and the Run Query by Definition actions on the HPE-UD system.

To create and configure a user in HPE-UD:

1. In the HPE-UD interface, click **Security**, **Roles Manager**.
2. Create a new role for FlexNet Manager import.
3. Select the role that you just created and click **General Actions** and assign the Run Query By Definition and the Run Legacy API actions to this role.
4. In the HPE-UD interface, click **Security**, **Users and Groups**.
5. Create a new user for FlexNet Manager import.
6. Click the **Roles** tab and assign the role that you created in Step 2 to this user.
7. Click the **Permissions Overview** tab and review the assigned permissions for this role.
8. Click **Data Flow Management**, **Universal Discovery**, **Zone-Based Discovery** and expand the **Mapping Options** section in **Preferences**.
9. Check the **Raw OS Installed Software** and set the **Include** option to `name=.*`. This enables HPE-UD adapter to

capture all raw installation evidence.

You have created and configured an HPE-UD user for use in the HPE-UD staging tool. You must add this user to the `FNMPuCMDBSettings.xml` file.

Installing and Configuring the Staging Tool

This procedure includes many separate sub-processes to complete the setup of the HPE-UD adapter.

To install and configure the HPE-UD adapter:

1. Ensure that the account under which the adapter executable will run has read/write/execute permissions on this `uCMDB_staging` database. Authentication may be through Windows authentication or SQL Server authentication. Using Windows authentication, the default account is the username running the `HPuCMDBStage.exe` adapter. The username and password for SQL Server authentication are specified in the database connection string, which you may supply in `FNMPuCMDBSettings.xml`, or override with the `-c` option on the command line.
2. To extract information from the HPE-UD system, the `HPuCMDBStage.exe` tool must have the required permissions to query the HPE-UD system. You must create a user in the HPE-UD system and assign it with permission to perform the Run Legacy API and the Run Query by Definition actions. You must specify this user in the `FNMPuCMDBSettings.xml` file. For more information on configuring the HPE-UD system, see Configuring HPE Universal Discovery System.
3. Navigate to the fixed location on the inventory beacon where the inventory reader must find configuration files to control its uploads. Verify the existence of the following path: `C:\ProgramData\Flexera Software\Compliance\ImportProcedures\Inventory\Reader\HP Universal Discovery`.
4. Create a folder to contain the adapter executable and its configuration file. Location is not critical; a suggested path is under `C:\Program Files\Flexera Software` (for version 2015 and later, or for a stand-alone server that is not an inventory beacon or application server). In your chosen location, create a folder such as `HPUDAdapter`.
5. From the `HPuCMDBStage\` folder within your unzipped archive, copy both `HPuCMDBStage.exe` and `FNMPuCMDBSettings.xml` to your newly created folder (such as `C:\Program Files\Flexera Software\HPUDAdapter`).
6. Open your copy of `FNMPuCMDBSettings.xml` in a text editor of choice, and review the self-documenting comments within that file. Modify the following values as required. Ensure that you specify the IP address of your HPE-UD server. You can use the command line options to configure `HPuCMDBStage.exe`. Use the '-h' option to get the list of available options.
   - Update values in the first element describing the downstream connection to the HPE-UD server, including the IP address, port, the account name and password for access. Keep a record of the account name and password for registering with HPE-UD. The default values are:

     ```
     <server protocol="http" address="10.200.20.138"
     port="8080" username="exportuser" password="Pa$$w0rd" timeout="3600"/>
     ```

Tip: *If you do not wish to record the password in the plain text configuration file, you can use a script to retrieve the password from an encrypted store, and supply it as a command-line option when starting the* `HPuCMDBStage.exe` *tool. Here is an example:*

```
HPuCMDBStage.exe -p <password>
```

- Update the second element for the connection to the staging database. The default values are:

```
<database
connection-string="Server=.;Database=uCMDB_Staging;Trusted_Connection=yes;"/>
```

- Update the third element to configure whether, and where, the executable should save XML files of the inventory collected from HPE-UD. The default value stores any XML files below the location of the executable: `<staging path="StagingData" method="stream"/>` You may wish to redirect the path setting for easier access for human inspection.
- Make sure that the user details in the `<Server>` section match with the user configured in the HPE-UD system.
- Save the settings file.

7. Validate the HPE-UD adapter operation. For more information about validating the adapter, see Validating the HPE-UD Adapter.
8. Assuming that you do not wish to trigger the adapter manually every time it needs to run, start Windows Task Scheduler, and create a basic task to run the adapter.

Important: *It is suggested that you schedule the adapter to run at times which cannot overlap with the inventory reader uploading the results to the application server. Check the schedule for the compliance reader on the central application server, and avoid this time slot.*

By default, the inventory import (starting with the reader) is triggered around 2 am. Therefore you might consider scheduling this task for some time such as 10 pm daily. The command line for the scheduled task (assuming that you have saved your preferred settings) is simply to invoke the executable. Any parameters not specified on the command line are taken from the settings file in the same folder as the executable. Here is an example:

```
HPuCMDBStage.exe -x <settings file>
```

This completes the configuration of the adapter executable.

3

Operation and Validation

This section describes the normal operation of HPE-UD adapter and also lists the steps to validate its operation. This section has the following tasks:

- HPE-UD Operation: See HPE-UD Adapter Operation
- Validating the HPE-UD Adapter: See Validating the HPE-UD Adapter

HPE-UD Adapter Operation

Normal operation of the HPE-UD adapter relies on the following sequence of events:

1. According to the scheduled instructions, the adapter reads the current content of the HPE-UD database and stages the new data in the staging database after removing the previously stored data. A flag stored in the staging database indicates the status of the data extraction. The resulting status is flagged within the database. For more information on installing and configuring the staging tool, see Installing and Configuring the Staging Tool
2. Following the schedule on the central application server, and provided that the staging status is `Success`, the import reader uploads this content to the inventory database.
3. The next compliance import brings the final data set into the compliance database, where it is automatically taken into account for compliance calculations. As always, for compliance calculations to proceed, the inventory records must be recognized by the Application Recognition Library, and you must have the resulting application records linked to the appropriate license.

Validating the HPE-UD Adapter

To validate the adapter operation:

1. Manually trigger the adapter executable.

   You can specify a value for the `method` parameter if you need to override the default settings set in the settings XML file. For example:

```
'C:\Program Files\Flexera Software\HPUD\HPuCMDBStage.exe' -f 'C:\temp' -m staged
```

This will write XML files under your `C:\temp` directory for review. It will also write data into the staging database.

2. Inspect the saved XML files to validate the inventory gathered.
3. Use SQL Server Management Studio to validate that the data is written to the staging database. Also review the `StagingState` property in the `uCMDBStagingDatabaseConfiguration` table in the staging database. Possible values are `Running`, `Failed`, or `Success`. This value must be `Success` before the HPE-UD data can be uploaded from the staging database to the central inventory database.
4. Wait until the next inventory import and calculations have run. You might have to wait overnight for this.
5. Use FlexNet Manager Suite to validate that new evidence has been recovered. Identify which evidence has been recognized by the ARL and which new rules are required. Link the applications to appropriate licenses.

IX

Managing Microsoft Office 365 Licenses

Microsoft Office 365 is a set of productivity tools available online (through a cloud-based subscription license) and offline (installed locally). Office 365 subscriptions include a range of productivity tools including Microsoft Office, Microsoft Exchange, Microsoft SharePoint, Microsoft Yammer, and Microsoft Skype for Business product lines. Multiple subscription plans (such as E1, E3, and E5) are available. Some of these plans allow you to install Office applications locally.

With detailed information about Microsoft 365, this chapter may help you in managing Microsoft 365 licenses through FlexNet Manager Suite.

1

Microsoft Office 365 License Management Considerations

Microsoft Office 365 licenses that allow local Office installations also allow you (subject to the chosen plan) to install Microsoft Office 365 on up to five PCs or Macs, and up to five tablets or smart phones, per `Named User` license. The following section provides an overview of the tools provided in each enterprise plan.

With the FlexNet Manager for Clients product, you can use FlexNet Manager Suite to manage your Office 365 subscriptions. You can:

- Validate that the desired users have the right plan
- Determine how many Office 365 licenses you need to purchase, while negotiating license agreements
- Ensure that business units are not under- or over-subscribed.

Based on the selected plan, a subscription of Microsoft Office 365 may allow you to access multiple Office applications online as well as offline (local installations) on up to five computers, and five tablets or smart phones. Because of this mixture, the compliance calculations for Microsoft Office 365 licenses are different than other applications.

For example, the *E3* plan allows you to install and use the following applications:

- Exchange Online Plan 2 for Office 365
- Skype for Business Online for Office 365
- Office Professional Plus for Office 365
- Azure Rights Management for Office 365
- SharePoint Online for Office 365
- Yammer for Office 365

Tip: *For a detailed description of the available Microsoft Office 365 subscription plans, and the applications supported in each plan, see the Microsoft Office 365 website.*

Considerations for managing Microsoft Office 365 licenses

To differentiate user records at various stages of processing, those already recorded within the compliance database of FlexNet Manager Suite are here called "compliance users".

- **Users**: Each user imported from Microsoft Office 365 is mapped to the appropriate compliance user record within FlexNet Manager Suite (by matching the **Email** or **Alternate email** field values). If an imported user cannot be mapped to any compliance user, a new compliance user record is created with a dummy inventory device assigned to it. These users consume against the Microsoft Office 365 multi-product license created for the Microsoft Office 365 tenant's subscription.

  Tip: *There are frequently differences between a user's corporate email address (such as `Sam.Doe@tmnis.com`) and the same user's email address registered in Microsoft Office 365 (such as `Sam.Doe@tmnisoffice365portal.com`). Since only the email addresses are available for matching (given that your Active Directory accounts are not reflected in the Microsoft Office 365 online service), this can lead to duplicated user records. To avoid duplicate records, you can, prior to importing from Microsoft Office 365 Online Service, populate your existing user records with the Microsoft Office 365 email address in the* ***Alternate email*** *field. You may want to consider using a Business Adapter to make widespread updates. If you have not done this prior to importing from the Microsoft Office 365 (and as a result some user records are now duplicated), you can update the* ***Alternate email*** *field in your AD-based records at any time, and the next full import and compliance calculation automatically cleans up those duplicates.*

- **Devices**: Although the installed device information is available to Office 365 administrators through the Office 365 Admin Center, FlexNet Manager Suite imports no device information from Microsoft Office 365. Instead, imported users are mapped to compliance users within FlexNet Manager Suite, and the inventory devices linked to those compliance users are considered to be the accessing devices. The method of linking the inventory devices and users may be any of assigned user, calculated user, or last logged on user (in descending order of priority, and all visible in the **Ownership** tab of the inventory device properties). The system also handles the following exceptions:
  - *Missing device:* If an imported user maps to a compliance user who is *not* linked to any inventory device, a dummy remote device is created by using the naming convention `User Name (Remote)`. Such a dummy device is not treated as a managed device by FlexNet Manager Suite (and specifically, does not count as a licensed device for the license for itself).
  - *Installations on extra devices:* If an installation of Office 365 is detected on a device, and the device is *not* linked to any compliance user matched to a user imported from Microsoft Office 365, the installation consumes from an existing perpetual license.

- **Mobile devices**: Because FlexNet Manager Suite does not collect inventory from all mobile devices (such as iPads), you may need to monitor these devices separately. You can get the details of installed devices from the Office 365 Admin Center. If this reveals too many devices where a single named user is authoring or editing Microsoft Office 365 documents, you may have to *deactivate* Microsoft Office 365 from one (or more) of the mobile devices.

  Tip: *You can read or present documents on a deactivated copy of Microsoft Office 365.*

- **Local installations**: A Microsoft Office 365 subscription is represented as a `Named User` license, with each user permitted to have installations on multiple devices. Only Microsoft manages the access to Microsoft Office 365. Separately, if imported inventory identifies a local installation related to Microsoft Office on an inventory device within your enterprise, the compliance user linked to the inventory device is checked. If this compliance user is

matched in the subscription for Office 365, FlexNet Manager Suite assumes that the software installation is licensed. If the device user is not matched in the subscription for Office 365, FlexNet Manager Suite looks for an existing perpetual license to cover this installation.

- **Office versions**: A perpetual license of Office 365 enables you to downgrade to an earlier version, but the subscription license does not. If you have a perpetual license of the older Office version, and you buy a subscription, the license validity of the older version gets extended until the subscription expiry date.
- **License entitlements**: The entitlements data for Microsoft Office 365 is directly imported from Microsoft Office 365. When an import matches an existing license record for Microsoft Office 365, its available entitlements are adjusted with each import. On the other hand, if no license exists for an imported subscription to Microsoft Office 365, a new Named User license is created for each subscription owned by each Microsoft Office 365 tenant, and the available entitlements are recorded there. In the license properties for this automatically-created license:
  - The **Duration** value (on the **Identification** tab) is set to `Subscription`.
  - Initially, no purchases are linked (on the **Purchase** tab), since your entitlements are imported from Microsoft Office 365. Should you wish to link purchases to this automatically-created license (for example, as a way of independently keeping track of your subscription purchases), notice that the value of the **Assigned** column is set to `0` (that is, the effective quantity on any linked purchases does *not* affect your total entitlements in the usual way). As well, the **Entitlement status** column displays `Not contributing`, to again remind you that for such a license, the compliance calculations are based entirely on the entitlement figures imported from Microsoft, rather than on purchase records.

2

Managing Office 365 Licenses through FlexNet Manager Suite

Here is a high-level procedure for managing Microsoft Office 365 licenses using FlexNet Manager Suite.

To manage Office 365 licenses:

1. To avoid duplicate user records, ensure that the user email addresses recorded in Microsoft Office 365 are saved in the user records visible in **Enterprise > All Users**.

   These may be called "compliance users" after the compliance database table where they are saved. The Microsoft Office 365 email addresses may be saved in either the **Email** field or the **Alternate email** field (both on the **Details** tab of the user properties). Common practice is to preserve the corporate email address in the first of these, and save the one from Microsoft Office 365 in the **Alternate email** field.

   Tip: *If you have many records to update, consider preparing a spreadsheet that lists your enterprise (Active Directory) email addresses against the Office 365 email addresses for each user, and creating a business adapter to update your records from the spreadsheet by inserting the Office 365 email addresses into the* `ComplianceUser.AlternateEmail` *column in your compliance database. Of course, if it is not practical to do this preparation before your first import from Microsoft Office 365, you can add the Office 365 email addresses to your AD-based user records at any time. Thereafter, the next full import and compliance calculation removes any duplicate user records, based on the now-matching email addresses.*

2. Ensure that you have created and scheduled a connector from FlexNet Manager Suite for each tenant of Microsoft Office 365.

   Typically, each enterprise is represented as a single tenant, although mergers and acquisitions may mean that your enterprise has multiple Microsoft tenants. For details on how to create and schedule a connector to Microsoft Office 365, see Creating Connections using the Microsoft 365 Connector.

   Note: *The* ***Microsoft 365*** *connector is the recommended connector to use to create a connection to Microsoft 365. However, if you would like instructions using the legacy* ***Microsoft Office 365 (deprecated)*** *connector, see Creating Connections using the Microsoft Office 365 (Deprecated) Connector.*

3. During the next discovery and inventory collection (following a scheduled connection and the upload of the

resulting data), FlexNet Manager Suite creates or updates the following objects:

- **Users**: Each imported user from Microsoft Office 365 is mapped to the appropriate compliance user record (by matching the **Email** or **Alternate email** field values). If an imported user cannot be mapped to any compliance user, a new compliance user record is created with a dummy inventory device assigned to this user.
- **Licenses**: A `Named User` multi-product license (*`MS-Tenant-name`* `Office 365 (Plan` *`XX`*) is created for each subscription plan linked to each Microsoft tenant:
  - This license is derived from a template provided in the downloadable libraries, and selected based on the subscription plan identifier.
  - The name of the Microsoft tenant is included as a prefix in the name of each of these licenses. This helps to distinguish licenses when you have multiple Microsoft Office Online Service tenants.
  - Multiple applications are linked to this license, as defined in the license template.

    Note: *If a particular plan is not currently supported by FlexNet Manager Suite, a license with no linked application is created. When the support for this missing plan is added during subsequent automatic updates to the SKU library and Application Recognition Library, a recommended license change appears, suggesting that you link the newly-identified applications to the license.*

  - On the **Use rights & rules** tab of the license properties, in the **License consumption rules** section, the **Allocations consume license entitlements** check box is set. This is because Microsoft Office 365 is a `Named User` license, and allocations to the users must consume entitlements.
  - On the **Consumption** tab of the license properties, the template sets the **Allocation type** to `Permanent`.

4. Review the **All Users** and **All licenses** pages to ensure that the required users and licenses have been created.
5. After license reconciliation has been completed, the **Product Summary** page shows the compliance status for Microsoft Office 365.
6. Navigate to **License Compliance > All Licenses** and filter the records to view Microsoft Office 365 licenses. You can check the **Consumed** and **Used** columns to know how many subscription entitlements are being used.
7. You can also view the following reports for insights on Microsoft Office 365 license consumption:
   - `User license details`
   - `License Overlap`.

   Note: *A subscription of Microsoft Office 365 enterprise plan (except E1) allows you to install Microsoft Office 365 on up to five PCs or Macs, and up to five tablets. FlexNet Manager Suite can only track the installation on devices that are present within your enterprise network. If you wish to include any installation outside the enterprise network, you can create a dummy inventory device and allocate a Microsoft Office 365 license to track the usage.*

Changes in subsequent imports

Each subsequent import from Microsoft Office 365 updates the following for each license of Microsoft Office 365:

- Entitlement count
- Expiry date
- Allocations correctly configured (as described above).

3

Connecting to Microsoft Office 365

This chapter provides information about how FlexNet Manager Suite connects to Microsoft Office 365.

Connector overview

There are currently two connectors available. It is recommended that you use the use the **Microsoft 365** connector, a replacement for the existing **Microsoft Office 365 (deprecated)** connector. To help during the transition to the **Microsoft 365** connector, the legacy connector will remain available for a period of time. The following table provides an overview comparison of the connectors. This table assumes you have installed the FlexNet Beacon released with FlexNet Manager Suite 2019 R1 or later.

Connector Name	Installation folder	Notes
Microsoft 365 ***Note:*** *This is the latest connector.*	`ProgramData\Flexera Software\Compliance\ImportProcedures\Inventory\Reader\Microsoft 365`	Note the following: • This Tier 1 inventory connector is now available to all customers. • The **Source Type** (on the inventory beacon) is **Microsoft 365** .
Microsoft Office 365 (deprecated) ***Note:*** *This is the legacy connector that is now deprecated.*	`ProgramData\Flexera Software\Compliance\ImportProcedures\Inventory\Reader\Microsoft Office 365` ***Note:*** *This installation folder name and path remains unchanged.*	Note the following: • This Tier 1 inventory connector remains available although deprecated and will longer be supported on a future release. • This connector's **Source Type** (specified on the inventory beacon PowerShell dialog) is **Microsoft Office 365 (deprecated)**. However, If you have not installed the FlexNet Beacon released with FlexNet Manager Suite 2019 R1 or later, then **Microsoft Office 365 (deprecated)** will not appear as a **Source Type** connection on the **Create PowerShell Source Connection** dialog, and instead will remain **Microsoft Office 365**.

Generating the Microsoft Office 365 license compliance position

To generate the compliance position for Microsoft Office 365 licenses, FlexNet Manager Suite needs information about subscriptions and usage of Microsoft Office 365 licenses (both online and offline):

- The local inventory collection process (within your enterprise) returns the local installations of Microsoft Office that are tied to your Microsoft Office 365 subscription(s).
- To get subscription and usage information from Microsoft Office Online Service, the FlexNet Beacon must be configured with an inventory connection of source type **Microsoft 365** or **Microsoft Office 365 (deprecated)** for each tenant of Microsoft Office 365. (Typically, each enterprise is a single Microsoft tenant; but your corporate history, including mergers and acquisitions, may mean that your enterprise includes multiple Microsoft tenants. Each tenant may then have one or more subscriptions to different licensing plans, such as Midsize Business or Enterprise.)

When you have created and configured the Microsoft Office 365 connection, the relevant inventory beacon imports the licenses, users, and usage information from the Office 365 Online Service account and uploads it to FlexNet Manager Suite according to your chosen schedule. For details on how to create and schedule a connector to Microsoft Office 365, see Creating Connections using the Microsoft 365 Connector or see Creating Connections using the Microsoft Office 365 (Deprecated) Connector.

Prerequisites and Configuration Considerations

This chapter describes prerequisites and configuration considerations for:

- Each inventory beacon that needs to download data from Microsoft Office 365
- The **Microsoft 365** or **Microsoft Office 365 (deprecated)** connector

FlexNet Beacon

The FlexNet Beacon released with FlexNet Manager Suite 2018 R2 or later is required to use the **Microsoft 365** connector. However, installing the FlexNet Beacon included in the 2019 R1 or later release provides maximum ease-of-use with the **Microsoft 365** connector by including auto-populated values on the **Create PowerShell Source Connection** dialog (that otherwise need to be entered manually). In addition, if you are using a using a FlexNet Beacon released prior to FlexNet Manager Suite 2019 R1, then **Microsoft Office 365 (deprecated)** will not appear as a **Source Type** connection on the **Create PowerShell Source Connection** dialog, and instead will remain **Microsoft Office 365.**

In addition, the inventory beacon that will collect inventory for Office 365 in the cloud requires a 64-bit operating system: Windows Server 2008 R2 SP1 or later, or Windows 7 SP1 or later.

Microsoft connector

Ensure that the account used to connect to the Microsoft Office 365 tenant(s) has the required privileges:

- For the **Microsoft 365** connector, the **Cloud application administrator** role is required in order for the FlexNet Beacon to retrieve a token that allows read only access to Microsoft Graph. For more information, see the note below as well as: https://docs.microsoft.com/en-us/azure/active-directory/users-groups-roles/directory-assign-admin-roles.

Note: *To be able to show Office 365 license consumption information in FlexNet Manager Suite, Microsoft requires read access for Users, Subscribed SKUs, and Reports from Office 365. FlexNet Beacon uses Microsoft Graph APIs to retrieve this data from Office 365. Microsoft Graph APIs only allow an administrator to read all of this information. However, FlexNet Beacon does not use credentials to connect to Office 365 via Microsoft Graph, which would give it full access as the user, but instead requests a token from Microsoft. Microsoft generates this token to allow access* ***only*** *to requested resources, and* ***only*** *for requested permissions. The token cannot be used to access any other resources and cannot be used to access any information on requested objects which requires other permissions. For example, a* ***User.ReadAll*** *permission only allows getting a list of users but does not allow getting a list of Calendar events or reading mail. The generated refresh token can only be used to access data that the user sees and consents to during the token generation process, which is offline read-only access to Active Directory and Reports (directory.****read****.all, reports.****read****.all, and offline_access). Offline means the FlexNet Beacon can connect and get data from Office 365 at schedule run without user actually signing in.*

- For the **Microsoft Office 365 (deprecated)** connector, if the maximum privileges of **Global administrator** cannot be used, then in order to collect usage data, the integration user must at a minimum have **Exchange administrator** and **Skype for Business administrator** roles in Office365 (available as check boxes under the **Custom administrator** role). For more information, see https://support.office.com/en-gb/article/About-Office-365-admin-roles-da585eea-f576-4f55-a1e0-87090b6aaa9d?ui=en-US&rs=en-GB&ad=GB.

Make sure that the following PowerShell connector prerequisite is met on each inventory beacon that needs to download data from Microsoft Office. These requirements should have been met when the inventory beacon was installed:

- PowerShell 5.1 or later, with the PowerShell execution policy set to `RemoteSigned`

Tip: *Run PowerShell with administrator rights to execute the following commands:*
To check the currently-installed version of PowerShell:

```
$PSVersionTable.PSVersion
```

To set PowerShell execution policy:

```
Set-ExecutionPolicy RemoteSigned
```

The following **Microsoft Office 365 (deprecated)** connector-specific prerequisites must be also met on each inventory beacon that needs to download data from Microsoft Office 365. These prerequisites are required for the **Microsoft Office 365 (deprecated)** connector only and are **not** needed for the **Microsoft 365** connector):

- 64-bit version of the Microsoft Online Services Sign-in Assistant (available from https://go.microsoft.com/fwlink/p/?LinkId=286152).
- Microsoft Azure Active Directory Module for Windows PowerShell. Microsoft Office 365 uses Azure Active Directory to manage user identities behind the scenes). To install the Microsoft Azure Active Directory Module for Windows PowerShell with these steps:
  1. Open an administrator-level PowerShell command prompt.
  2. Run the `Install-Module MSOnline` command.
  3. If prompted to install the NuGet provider, type `Y` and press `ENTER`.
  4. If prompted that the installer is not signed, type `Y` and press `ENTER`

5. If prompted to install the module from PSGallery, type Y and press ENTER.

- Skype for Business Online, Windows PowerShell Module 64-bit version (.

Creating Connections using the Microsoft 365 Connector

Important: *The instructions in this section are for use with the* ***Microsoft 365*** *connector. This is the recommended connector to use to create a connection to Microsoft Office 365 on an inventory beacon. However, if you would like instructions using the legacy* ***Microsoft Office 365 (deprecated)*** *connector, see Creating Connections using the Microsoft Office 365 (Deprecated) Connector.*

Use the following procedure to create a connection to Microsoft Office 365 on an inventory beacon using the **Microsoft 365** connector. A separate connection is required for each tenant of Microsoft Office 365. (Typically, there is one tenant per enterprise; but your corporate history, particularly of mergers and acquisitions, may mean that your enterprise has multiple Microsoft tenants.) The inventory beacon requires this connection to import entitlements, users, and usage information from the Microsoft Office 365 online account. Each per-tenant import covers all subscriptions for that tenant.

When you use the **Microsoft 365** connector, there are two ways of creating connections to the Microsoft Office 365 online service. Refer to the following sections for associated instructions:

- Using FlexNet Manager Suite's Multi-Tenant App to Connect to Microsoft 365 — The **Microsoft 365** connector uses a pre-registered multi-tenant app that has been configured by Flexera for the purpose of collecting Microsoft Office 365 data. This app is configured to allow customers to collect data without the overhead required to register an app using the Azure portal in their own Microsoft Office 365 instance.
- Registering an app using the Azure portal to connect to Microsoft 365 — This second method registers an app within your own Microsoft Office 365 instance.

Using FlexNet Manager Suite's Multi-Tenant App to Connect to Microsoft 365

Important: *The instructions in this section are for use with the* ***Microsoft 365*** *connector. This is the recommended connector to use to create a connection to Microsoft Office 365 on an inventory beacon. If you would like instructions using the legacy* ***Microsoft Office 365 (deprecated)*** *connector, see Creating Connections using the Microsoft Office 365 (Deprecated) Connector.*

To create a connection to Microsoft 365 on an inventory beacon using FlexNet Manager Suite's multi-tenant app:

1. Log into the inventory beacon interface as an administrator (for example, in the Windows Start menu, search for **FlexNet Beacon**, right-click it, and select **Run as administrator**).

Tip: *Remember that you must run the inventory beacon software with administrator privileges.*

2. From the **Data collection** group in the navigation bar, choose **Inventory Systems**.
3. Choose either of the following:
   - To change the settings for a previously-defined connection, select that connection from the list, and click **Edit...**.
   - To create a new connection, click the down arrow on the right of the **New** split button, and choose **Powershell**.
4. Complete (or modify) the values for the following required fields:
   - **Connection Name**: The name of the inventory connection. The name may contain alphanumeric characters, underscores or spaces, but must start with either a letter or a number. When the data import through this connection is executed, the data import task name is same as the connection name.
   - **Source Type**: Select **Microsoft 365** from this list.
5. Optionally, if your enterprise uses a proxy server to enable Internet access, complete (or modify) the values in the **Proxy Settings** section of the dialog box in order to configure the proxy server connection.
   - **Use Proxy**: Select this checkbox if your enterprise uses a proxy server to enable Internet access. Complete the additional fields in the **Proxy Settings** section, as needed. If the **Use Proxy** checkbox is not selected, the remaining fields in the **Proxy Settings** section are disabled.
   - **Proxy Server**: Enter the address of the proxy server using HTTP, HTTPS, or an IP address. Use the format `https://ProxyServerURL:PortNumber`, `http://ProxyServerURL:PortNumber`, or `IPAddress:PortNumber)`. This field is enabled when the **Use Proxy** checkbox is selected.
   - **Username** and **Password**: If your enterprise is using an authenticated proxy, specify the username and password of an account that has credentials to access the proxy server that is specified in the **Proxy Server** field. These fields are enabled when the **Use Proxy** checkbox is selected.
6. In the **Microsoft 365** section, do the following:
   a. Note that the following fields are auto-populated if you have installed the FlexNet Beacon released with FlexNet Manager Suite 2019 R1 or later. If you have not installed this FlexNet Beacon, then manually enter the following values into these fields:
      - **Authorization Endpoint**: `https://login.microsoftonline.com/common/oauth2/v2.0/authorize`
      - **Token Endpoint**: `https://login.microsoftonline.com/common/oauth2/v2.0/token`
      - **Application (client) ID**: `5bb1a5a2-0d97-4335-9448-119f7b27aff9`
      - **Redirect URI**: `https://login.microsoftonline.com/common/oauth2/nativeclient`

   b. Next to the **Refresh Token** click the **Generate** button to generate a refresh token that will be used to integrate with Microsoft 365.

      When you click the **Generate** button to the right of the **Refresh Token** field, a **Microsoft** popup appears asking you to **Pick an account** to use to log into Microsoft Office 365.

c. Choose an Active Directory account with **Cloud application administrator** role privileges and enter the password.

> ***Tip:*** *The **Cloud application administrator** role is required in order for the FlexNet Beacon to retrieve a token that allows read only access to Microsoft Graph. For more information, see https://docs.microsoft.com/en-us/azure/active-directory/users-groups-roles/directory-assign-admin-roles. The generated refresh token can only be used to access data that user sees and consents to during the token generation process, which is offline read-only access to Active Directory and Reports (directory.**read**.all, reports.**read**.all, and offline_access). Offline means the FlexNet Beacon can connect and get data from Office 365 at schedule run without user actually signing in.*

A **Permissions requested** dialog appears.

d. Click **Consent on behalf of your organization** to accept the read only permissions that will be granted to the refresh token.

e. Click **Accept**.

The **Refresh Token** field is now populated.

7. At the bottom of the **FlexNet Beacon** interface, click **Save**, and if you are done, also click **Exit**.

> ***Tip:*** *Consider whether you want to select your connection, and click **Execute Now**, before you exit. For information about scheduling data imports through this connection, see Scheduling a Connection in the online help.*

After a successful data import, the users, applications, licenses, and usage data are all visible in the appropriate pages of FlexNet Manager Suite.

> ***Note:*** *To know more about the operations available on the **Inventory Systems** page of FlexNet Beacon, see Inventory System Page in the online help. For scheduling data imports through this connection, see Scheduling a Connection, also in help.*

Registering an app using the Azure portal to connect to Microsoft 365

> ***Important:*** *The instructions in this section are for use with the **Microsoft 365** connector. This is the recommended connector to use to create a connection to Microsoft Office 365 on an inventory beacon. If you would like instructions using the legacy **Microsoft Office 365 (deprecated)** connector, see Creating Connections using the Microsoft Office 365 (Deprecated) Connector.*

An alternate method to the instructions provided in Using FlexNet Manager Suite's Multi-Tenant App to Connect to Microsoft 365 is to register an app in the Azure portal to connect to Microsoft 365 as shown in the steps in this section. This requires an Azure account with Global Administrator privileges.

To register an app in the Azure portal to connect to Microsoft 365:

1. With an Azure account with Global Administrator privileges, login to the Microsoft Azure portal.
2. In the left navigation bar, click **Azure Active Directory**.
3. Click **App registration**.
4. Click the **New registration** button.

   The **Register an application** page appears.
5. In the **Register an application** page, do the following:
   a. In the **Name** field, enter `FlexNet Beacon`.
   b. In the **Supported account types** section, choose **Accounts in this organizational directory only**.
   c. Click **Register**.

      The **FlexNet Beacon** page appears.
6. On the FlexNet Beacon page, click **Authentication** in the middle navigation bar and do the following.
   a. In the **Redirect URIs** section, enter **FlexNet Beacon**.
   b. In the **Suggested redirect URIs for public clients (mobile, desktop)** section choose the redirect you would like to use, such as **https://login.microsoftonline.com/common/oauth2/nativeclient**.
   c. Click **Save**. After a period of time, the registration completes.
7. The **FlexNet Beacon** page appears after the registration is complete. Keep this browser and Azure session open to the **FlexNet Beacon** page so that you can copy and paste values from it to fields in the **FlexNet Beacon Create PowerShell Source Connection** dialog in Step 12.
8. Log into the inventory beacon interface as an administrator (for example, in the Windows Start menu, search for **FlexNet Beacon**, right-click it, and select **Run as administrator**).

   Tip: *You must run the inventory beacon software with administrator privileges.*
9. To create a new connection, click the down arrow on the right of the **New** split button, and choose **PowerShell**.
10. Complete the values for the following required fields:
    - **Connection Name**: Enter the name of the inventory connection. The name may contain alphanumericcharacters, underscores or spaces, but must start with either a letter or a number. When the data import through this connection is executed, the data import task name is same as the connection name.
    - **Source Type**: Select **Microsoft 365** from this list.
11. Optionally, if your enterprise uses a proxy server to enable Internet access, complete the values in the **Proxy Settings** section of the dialog box in order to configure the proxy server connection:
    - **Use Proxy**: Select this checkbox if your enterprise uses a proxy server to enable Internet access. Complete the additional fields in the **Proxy Settings** section, as needed. If the **Use Proxy** checkbox is not selected, the remaining fields in the **Proxy Settings** section are disabled.
    - **Proxy Server**: Enter the address of the proxy server using HTTP, HTTPS, or an IP address. Use the format

https://ProxyServerURL:PortNumber, http://ProxyServerURL:PortNumber, or IPAddress:PortNumber). This field is enabled when the **Use Proxy** checkbox is selected.

- **Username and Password**: If your enterprise is using an authenticated proxy, specify the username and password of an account that has credentials to access the proxy server that is specified in the **Proxy Server** field. These fields are enabled when the **Use Proxy** checkbox is selected.

12. To enter the fields in the **Microsoft 365** section, you need to go back to your Azure session, opened to the **FlexNet Beacon** page. The fields in Azure have a **Click to copy** icon that lets you copy their values to the clipboard for pasting into the following **FlexNet Beacon Create PowerShell Source Connection** dialog:
    - **Authorization Endpoint**: Paste this value from the Azure session. (In Azure, click **Overview** and then click **Endpoints**. Click the **Click to copy** icon to the right of the **OAuth 2.0 authorization endpoint** field.)
    - **Token Endpoint**: Paste this value from the Azure session. (In Azure, click **Overview** and then click **Endpoints**. Click the **Click to copy** icon to the right of the **OAuth 2.0 token endpoint** field.)
    - **Application (client) ID**: Paste this value from the Azure session. (In Azure, click **Overview** and then click the **Click to copy** icon to the right of the **Application (client) ID** field.)
    - **Redirect URI**: Paste this value from the Azure session. (In Azure, click **Overview** and then click the hyperlink in the **Redirect URIs** section. This opens the **Redirect URIs** page. Click the **Click to copy** icon to the right of the**Redirect URI** setting you selected or if you specified a custom URI copy that URI.)
13. Click the **Generate** button to generate a refresh token that will be used to authenticate the connection to Microsoft 365.
14. At the bottom of the **FlexNet Beacon** interface, click **Save**, and if you are done, also click **Exit**.

Tip: *Consider whether you want to select your connection, and click* ***Execute Now****, before you exit. For information about scheduling data imports through this connection, see Scheduling a Connection in the online help.*

After a successful data import, the users, applications, licenses, and usage data are all visible in the appropriate pages of FlexNet Manager Suite.

Note: *To know more about the operations available on the* ***Inventory Systems*** *page of FlexNet Beacon, see Inventory System Page in the online help. For scheduling data imports through this connection, see Scheduling a Connection, also in help.*

Configuring Token Lifetimes in Azure Active Directory

This section is for Microsoft Azure AD administrators who may want to configure the lifetimes of refresh tokens and access tokens issued by Azure Active Directory. If your organization already have these set, these steps are not necessary.

FlexNet Beacon uses an Microsoft Azure Active Directory (AAD) native app for authentication when using Microsoft 365 inventory connections. A customer has an option to use the Flexera-created native app for this authentication or they can create their own.

When the authentication is complete and a user consents to access to the resource (Microsoft Graph) with read only permissions, the Azure AD generates and sends two tokens: a refresh token and an access token. These tokens are specific to the user, resource, and permissions. The refresh token is used to authenticate further in the future without a need to login while the access token is a session token. Typically, a refresh token is saved and is used first in every session to generate a new access token, once the access token is generated, it is then used in following calls within that session.

Since these tokens can be used anytime without a need for a user to manually login, Azure AD allows to configure the lifetime for such tokens. After a refresh token is expired, a user must login and consent access to resource and permissions to get a new refresh token generated. After an access token is expired, an app can use a valid refresh token to get a new access token.

The configuration of these tokens lifetime is an Azure AD functionality and is applied to all applications in that tenant. To configure these tokens, an Azure AD administrator must have the Azure AD PowerShell module installed. For more information these tokens, their default values and configuration, see https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-configurable-token-lifetimes.

To configure the refresh token and an access token lifetimes, an Azure AD administrator opens an elevated PowerShell prompt (run as administrator) and performs the following PowerShell commands (shown in codeboxes in these steps):

1. Install the Azure AD PowerShell module:

```
Install-Module AzureADPreview
```

2. Connect to Azure AD:

```
Connect-AzureAD
```

3. Check if you already have a token lifetime policy:

```
$defaultTokenPolicy = Get-AzureADPolicy | Where-Object {$_.Type -eq
"TokenLifetimePolicy" -and
$_.IsOrganizationDefault -eq $true}
```

4. Check whether a token policy exists:

```
$defaultTokenPolicy
```

5. If a token policy exists, the previous command returns an object; otherwise, a blank. If a value is returned, you may want to examine the current token policies by entering the following:.

```
$defaultTokenPolicy.Definition
```

6. If a policy exists, it is returned. The following shows an example policy returned:

```
PS C:\WINDOWS\system32> $defaultTokenPolicy.Definition
{
    "TokenLifetimePolicy":
    {
        "Version":1,
```

```
        "AccessTokenLifetime":"0.00:10:00",
        "MaxInactiveTime":"90.00:00:00",
        "MaxAgeSingleFactor":"until-revoked",
        "MaxAgeMultiFactor":"until-revoked",
        "MaxAgeSessionSingleFactor":"until-revoked",
        "MaxAgeSessionMultiFactor":"until-revoked"
    }
}
```

The `AccessTokenLifetime` in the above example is set to 10 minutes which means that once an access token is generated, it remains active for ten minutes, after which the app must retrieve another generated access token. The `MaxInactiveTime` is set to 90 days which means that a refresh token expires after 90 days of inactivity. The `MaxAgeSingleFactor` and `MaxAgeMultiFactor` are also related to refresh token and define the maximum lifetime of a refresh token, based on the single or multi-factor authentication setting of your organization.

7. If you want to add or update your Azure AD token lifetime settings, you need to decide on the new settings and execute following (updating the lifetimes as you wish):

```
$newTokenPolicy = @('{
    "TokenLifetimePolicy":
    {
        "Version":1,
        "AccessTokenLifetime":"0.01:00:00",
        "MaxInactiveTime":"90.00:00:00",
        "MaxAgeSingleFactor":"until-revoked",
        "MaxAgeMultiFactor":"until-revoked",
        "MaxAgeSessionSingleFactor":"until-revoked",
        "MaxAgeSessionMultiFactor":"until-revoked"
    }
}’)
```

8. Now, if you did not have a token policy, execute the following.

```
New-AzureADPolicy -Type "TokenLifetimePolicy" -DisplayName
"OrganizationDefaultPolicyScenario"
-IsOrganizationDefault $true -Definition $newTokenPolicy
```

9. And if you had a token policy, execute the following cmd to update it.

```
Set-AzureADPolicy -Id $defaultTokenPolicy.Id -DisplayName
"OrganizationDefaultPolicyUpdatedScenario"
-Definition $newTokenPolicy
```

10. To validate that the policy has been applied correctly, execute **step 3** through **5**.

Creating Connections using the Microsoft

Office 365 (Deprecated) Connector

Important: *The instructions in this section are for creating a connection to Microsoft Office 365 using the legacy* ***Microsoft Office 365 (deprecated)*** *connector. However, the* ***Microsoft 365*** *connector is recommended connector to use to create a connection to Microsoft Office 365 on an inventory beacon. For those instructions, refer to Creating Connections using the Microsoft 365 Connector.*

Use the following procedure to create a connection to Microsoft Office 365 on an inventory beacon using the legacy **Microsoft Office 365 (deprecated)** connector. A separate connection is required for each tenant of Microsoft Office 365. (Typically, there is one tenant per enterprise; but your corporate history, particularly of mergers and acquisitions, may mean that your enterprise has multiple Microsoft tenants.) The inventory beacon requires this connection to import entitlements, users, and usage information from the Microsoft Office 365 online account. Each per-tenant import covers all subscriptions for that tenant.

To create a connection to Microsoft Office 365:

1. Ensure that you have your preferred schedule for imports from Microsoft Office 365 set on the appropriate inventory beacon:
   a. Log into the inventory beacon interface as an administrator (for example, in the Windows Start menu, search for **FlexNet Beacon**, right-click it, and select **Run as administrator**).

      Tip: *Remember that you must run the inventory beacon software with administrator privileges.*

   b. From the **Data collection** group in the navigation bar, choose **Scheduling**.
   c. If there is not already a suitable schedule in the list, click **New...** and complete the details. Otherwise, identify the schedule you will use.
2. Select the **Inventory Systems** page (in the same navigation group).
3. Choose either of the following:
   - To change the settings for a previously-defined connection, select that connection from the list, and click **Edit...**.
   - To create a new connection, click the down arrow on the right of the **New** split button, and choose **Powershell**.
4. Complete (or modify) the values for the following required fields:
   - **Connection Name**: The name of the inventory connection. When the data import through this connection is executed, the data import task name is same as the connection name.
   - **Source Type**: Select `Microsoft Office 365 (deprecated)` from this list.

   Note: *If you have not installed the FlexNet Beacon released with FlexNet Manager Suite 2019 R1 or later, then* ***Microsoft Office 365 (deprecated)*** *will not appear as a* ***Source Type*** *connection. Instead,* ***Microsoft Office 365*** *will appear as the legacy connector source type.*

5. Optionally, if your enterprise uses a proxy server to enable Internet access, complete (or modify) the values in the **Proxy Settings** section of the dialog box in order to configure the proxy server connection.
   - **Use Proxy**: Select this checkbox if your enterprise uses a proxy server to enable Internet access. Complete the additional fields in the **Proxy Settings** section, as needed. If the **Use Proxy** checkbox is not selected, the remaining fields in the **Proxy Settings** section are disabled.
   - **Proxy Server**: Enter the address of the proxy server using HTTP, HTTPS, or an IP address. Use the format `https://ProxyServerURL:PortNumber`, `http://ProxyServerURL:PortNumber`, or `IPAddress:PortNumber)`. This field is enabled when the **Use Proxy** checkbox is selected.
   - **Username** and **Password**: If your enterprise is using an authenticated proxy, specify the username and password of an account that has credentials to access the proxy server that is specified in the **Proxy Server** field. These fields are enabled when the **Use Proxy** checkbox is selected.
6. Complete (or modify) the values in the **Microsoft Office 365** section of the dialog. All of the following values are required:

   Note: *If you have multiple tenants within Microsoft Office 365 (for example, separate subscriptions for different corporate units or locations), you need to create a separate connector for each tenant using its own credentials. Within each tenant, a single connection and a single import recovers data for all your subscriptions (if you have multiple subscriptions).*

   - **Username**: Microsoft Office 365 tenant user name.
   - **Password**: Microsoft Office 365 tenant password
7. **Save** the connection.
8. Select your new connection from the displayed list, and click **Schedule...**.
9. In the dialog that appears, select the name of your chosen schedule for inventory collection through this connection, and click **OK**.
10. At the bottom of the **FlexNet Beacon** interface, click **Save**, and if you are done, also click **Exit**.

    Tip: *Consider whether you want to select your connection, and click **Execute Now**, before you exit.*

After a successful data import, the users, applications, licenses, and usage data are all visible in the appropriate pages of FlexNet Manager Suite.

Note: *To know more about the operations available on the **Inventory Systems** page of FlexNet Beacon, see Inventory System Page in the online help. For scheduling data imports through this connection, see Scheduling a Connection, also in help.*

Troubleshooting Imports from Office 365

The following sections provide help with troubleshooting imports from Microsoft Office 365:

- Managing Office 365 Licenses through FlexNet Manager Suite

- Troubleshooting Microsoft Office 365 (Deprecated) Connector Imports from Office 365

Troubleshooting Microsoft 365 Connector Imports from Office 365

Important: *The troubleshooting in this section applies to connections to Microsoft Office 365 that use the* ***Microsoft 365*** *connector. For help with troubleshooting connections using the like instructions using the* ***Microsoft Office 365 (deprecated)*** *connector, see Troubleshooting Microsoft Office 365 (Deprecated) Connector Imports from Office 365.*

Symptom	Recommendation
The **Authorization Endpoint**, **Token Endpoint**, **Application (client) ID** or **Redirect URI** values are blank in the PowerShell connection dialog.	Check whether you are using a FlexNet Beacon older than the one included with FlexNet Manager Suite 2019 R1.. If so, you will need to manually enter these values: • **Authorization Endpoint**: `https://login.microsoftonline.com/common/oauth2/v2.0/authorize` • **Token Endpoint**: `https://login.microsoftonline.com/common/oauth2/v2.0/token` • **Application (client) ID**: `5bb1a5a2-0d97-4335-9448-119f7b27aff9` • **Redirect URI**: `https://login.microsoftonline.com/common/oauth2/nativeclient`
A blank pop-up screen appears when you click the **Generate** button (next to the **Refresh Token** field) when attempting to generate a refresh token that will be used to integrate with Microsoft 365.	You most likely need to set the PowerShell execution policy. Run PowerShell with administrator rights to execute the following command: `Set-ExecutionPolicy RemoteSigned`
You get a **Need admin approval** dialog that informs that the FlexNet Beacon needs permission to access resources in your organization that only an admin can grant.	Ensure that the account used to connect to the Microsoft Office 365 tenant(s) has the **Cloud application administrator** role. This role is required in order for the FlexNet Beacon to retrieve a token that allows read only access to Microsoft Graph. For more information, see https://docs.microsoft.com/en-us/azure/active-directory/users-groups-roles/directory-assign-admin-roles.

Symptom	Recommendation
You receive an authorization error that informs that you are not authorized to access this site.	This may be because the machines do not have permissions to access the Microsoft Authentication site. Ensure that you provide proxy information if the machine uses proxy settings. Also, open a Web browser, navigate tohttps://login.microsoftonline.com and log on when asked to validate you can successfully authenticate on this machine. If you still see issues, contact Flexera with information from this and other steps. Note the steps that caused your issue and save any necessary screenshots to report to Flexera. Also include what kind of authentication service is used in your organization and what region and country the machine and Office 365 tenant reside. Note that some regions (e.g., Germany) have separate login URLs as explained https://docs.microsoft.com/en-us/azure/active-directory/develop/authentication-national-cloud.

Troubleshooting Microsoft Office 365 (Deprecated) Connector Imports from Office 365

Important: *The troubleshooting in this section applies to connections to Microsoft Office 365 using the* ***Microsoft Office 365 (deprecated)*** *connector. For help with troubleshooting connections using the like instructions using the* ***Microsoft Office 365*** *connector, see Managing Office 365 Licenses through FlexNet Manager Suite.*

This process traces the path of data imported from Microsoft Office 365 Online Service through an inventory beacon to the central application server of FlexNet Manager Suite. You may pick up at any stage of the process as needed for your particular symptoms.

To troubleshoot imports from Microsoft Office 365 Online Service:

1. Validate that you have licensed the FlexNet Manager for Datacenters product:

   a. Navigate to the system menu (⚙ ▾ in the top right corner) in the web interface for FlexNet Manager Suite, and click **FlexNet Manager Suite License**.

   b. In the **Licensed products** section, scroll down to see the card for FlexNet Manager for Datacenters.

   Inventory from a connection to Microsoft Office 365 Online Service cannot be imported or uploaded by an inventory beacon without this license term being registered on the central application server. If it is missing, please contact your Flexera representative for assistance.

2. Validate imports to the inventory beacon:

   a. On the inventory beacon that connects to Microsoft 365, log into FlexNet Beacon using an account with administrator privileges.

   b. On the **Inventory Systems** page, select your connection to Office 365, and click **Execute Now**

      A confirmation shows that the extract has started. A typical import may take 2-3 minutes.

c. In the very top-left corner of the FlexNet Beacon interface, right-click the menu icon, and select **Open log file folder**.

   File Explorer opens.

d. Ensure that you are viewing the `ProgramData\Flexera Software\Compliance\Logging\ComplianceReader` folder.

e. In Notepad, open the text document `importer-[nnnn]` that was modified on today's date, and scroll to the bottom.

   If the reading of data is still in progress, exit from Notepad, and try again in a few minutes.

f. A successful process shows the following at the end of the log file (with each line prefixed with the date and time):

```
[INFO] 0 source data warnings
[INFO] 0 errors, 0 warnings
[INFO] Import has been completed successfully
```

   If instead there are warnings or errors in the log file, try to correct the issues. If all else fails, save the log file to include with a Support ticket for Flexera. Typical issues at this stage may include:

   - Proxy or firewall settings preventing access to the Internet
   - Intermittent issues with the Internet or access to the Office 365 portal
   - Incorrect details for the account name or password accessing the portal (see Creating Connections using the Microsoft Office 365 (Deprecated) Connector for details of configuring this account)
   - The connection has not been regularly scheduled (see the same topic), and so succeeds only when executed manually.

   Immediately after a successful import is completed, the inventory beacon attempts to upload the archive of imported data to its parent (the parent may be another inventory beacon in the hierarchy, or the central application server):

   - When the upload succeeds, the inventory beacon also saves a copy in `C:\ProgramData\Flexera Software\Beacon\IntermediateData\Uploaded`, and preserves this copy for 14 days
   - If the upload fails, the imported archive stays in `C:\ProgramData\Flexera Software\Beacon\IntermediateData`.

   Tip: *If you are working on a disconnected inventory beacon (one that cannot access the central application server), this `IntermediateData` folder contains the data files that you must transfer to another location where uploads are possible.*

g. To review the data imported from Microsoft 365, navigate to `C:\ProgramData\Flexera Software\Beacon\IntermediateData\Uploaded`, and inspect the most recent relevant `.zip` file there.

   Tip: *If the zip archive remains in the `IntermediateData` folder (on a connected inventory beacon), there is an upload problem, as discussed below.*

3. To debug uploads from this inventory beacon to its parent, review `C:\ProgramData\Flexera Software\`

`Compliance\Logging\ComplianceUpload\upload.log` for details of any networking issues, and note the URL of the next server in the upload chain. If this is another inventory beacon, repeat these reviews there, and so on until you exhaust the hierarchy of inventory beacons.

The upload URL is in a log line that starts:

```
dateTime [Upload.UploadProcess  ] [INFO] Uploading to http...
```

4. Once you have reached your application server (or inventory server, in a multi-server implementation), check `C:\ProgramData\Flexera Software\Compliance\Logging\WebResolvers\dispatcher.log` for any issues in resolving the inventory.
5. If you are unable to resolve problems, collect your logs together in a zip archive. Ask your registered support contact (a designated person within your enterprise who has access rights and login details) to open a new support case at https://community.flexera.com/t5/forums/postpage/board-id/@support, including a clear description of the issue. Once the case has been saved, your support contact can use the **Upload** button (in the **Attachments** section at the bottom) to attach your prepared zip archive of logs.

4

Migrating to a New Microsoft Connector

Important: *The instructions in this section are only applicable if you have previously used the* ***Microsoft Office 365*** *(or* ***Microsoft Office 365 (deprecated)*** *connector), have validated the data coming from the new* ***Microsoft 365*** *connector, and are ready to switch over completely to the new connector, There are a few basic steps to follow to migrate licenses and relink contracts and purchase orders to the new license .*

To migrate from an older to newer Microsoft connector, migrate licenses, and relink contracts and purchase orders to the new license:

1. Check your version of the FlexNet Beacon.

   Note: *The FlexNet Beacon released with FlexNet Manager Suite 2018 R2 or later is required to use the* ***Microsoft 365*** *connector. However, installing the FlexNet Beacon included in the 2019 R1 or later release provides maximum ease-of-use with the* ***Microsoft 365*** *connector by including auto-populated values on the* ***Create PowerShell Source Connection*** *dialog (that otherwise need to be entered manually). In addition, if you are using a using a FlexNet Beacon released prior to FlexNet Manager Suite 2019 R1, then* ***Microsoft Office 365 (deprecated)*** *will not appear as a* ***Source Type*** *connection on the* ***Create PowerShell Source Connection*** *dialog, and instead will remain* ***Microsoft Office 365.***

2. Delete the connection to the old **Microsoft Office 365** or **Microsoft Office 365 (deprecated)** connector.

   Note: *Although the legacy connector name is now* ***Microsoft Office 365 (deprecated)****, the previous name of the this connector's* ***Source Type*** *was* ***Microsoft Office 365****.*

3. Relink contracts and purchase orders to the new license.

X

Oracle Enterprise Manager Adapter

Oracle Enterprise Manager monitors and manages other Oracle software installed on customer sites. Therefore it is a useful aid in gathering inventory from Oracle systems.

The Oracle Enterprise Manager (OEM) adapter, from Flexera, connects to Oracle Enterprise Manager, and extracts a file of connection information for the Oracle systems it monitors. This file is in a standard format (`TNSNames.ora`) used by Oracle. (In fact, if you already have files of the same name generated by Oracle, you can simply copy these to the appropriate folder on an inventory beacon. This may be a viable alternative to using this adapter.)

When the file is saved to a special location on an inventory beacon (and the inventory rules for this inventory beacon allow processing of `TNSNames.ora` files), the connection information in it can be used by FlexNet Manager for Datacenters, a separately-licensed product within FlexNet Manager Suite, to collect software inventory information. This document covers the set up and configuration to use the adapter as part of an Oracle inventory solution.

Note: *The sole purpose of the OEM adapter is to prepare a `TNSNames.ora` file as a method of discovery of Oracle Database servers. There is a completely unrelated process where the normal FlexNet inventory of the database instance used by Oracle Enterprise Manager (the* ***OEM repository****) also reveals the names of the Oracle database instances it manages, as well as any Oracle options that have been enabled through the related console for Oracle Enterprise Manager. This latter process requires neither an adapter nor any special configuration; it is a natural by-product of gathering inventory from the* ***OEM repository****, whether by a locally installed FlexNet inventory agent, by direct inventory collection through an inventory beacon, or by any other means of gathering FlexNet inventory from the* ***OEM repository*** *database instance.*

Terminology

Throughout, the Flexera adapter is referred to as the **OEM adapter**. The database for Oracle Enterprise Manager (to which the OEM adapter connects) is referred to as the **OEM repository**.

The OEM adapter is tested for use with Oracle Enterprise Manager releases 12.1–13.3.

1

Understanding the Oracle Enterprise Manager Adapter

As well as providing a functional overview of the straight-forward OEM adapter itself, this chapter provides additional background about the other aspects of your system that collaborate to provide Oracle introspection and inventory reporting. As well, this chapter covers the prerequisites, content, and delivery of the OEM adapter.

How the Adapter Assists in Inventory Gathering

The OEM adapter provides an alternative or additional method of discovering Oracle Database servers in your computing estate. Discovery (by one means or another) is a prerequisite for collecting software inventory from the Oracle servers.

The OEM adapter is installed on a convenient computer that has network access to Oracle Enterprise Manager and its OEM repository. Typically, this computer may be a FlexNet Beacon. Multiple instances of the OEM adapter may be installed (on one or more computers), each of which can connect with one instance of Oracle Enterprise Manager.

Numbers in the diagram (where "OEMA" identifies the OEM adapter) correspond to the process description below:

The process runs as follows:

1. The Windows scheduled task triggers the OEM adapter, which contacts its instance of Oracle Enterprise Manager, and collects the connection data from the OEM repository.
2. The OEM adapter formats the data into a `TNSNames.ora` file, and saves it to a special location on the inventory beacon.

   You may configure the name of the saved file, and, if multiple instances of the OEM adapter save files to the same inventory beacon, you *must* modify the file names so that the files do not overwrite each other.

   Note: *This completes the function of the OEM adapter. The remainder of the process is standard operating procedure for FlexNet Manager Suite with the FlexNet Manager for Datacenters product installed. If you already have an operational system, several of these steps may already be completed.*

3. You set up a special account with read-only permissions on your Oracle Database for all the tables and views needed for Oracle introspection. One helpful practice is to use the same purpose-driven set of credentials on all servers. A utility is available to help created the account(s) required with the correct permissions.
4. The same credentials must be recorded in the Password Manager on each of the relevant inventory beacons.
5. In the web interface for FlexNet Manager Suite, you define a subnet (or several subnets) that contain the Oracle Database servers of interest; and you assign to these subnet(s) the inventory beacon(s) where the `TNSNames.ora` files are being saved. (Assignments are distributed automatically to inventory beacon(s), along with rules.)
6. Continuing n the web interface, you create a rule with an action including Oracle inventory collection, with the option to use TNS name file selected. The rule also has a target which matches the subnet(s) you are interested in. This rule is automatically distributed to inventory beacon(s).
7. On the inventory beacon(s) of interest:
   a. The rule and assignment are received.
   b. The inventory beacon assesses these, concludes that it is authorized to act, and looks for a any `*.ora` file(s) in the special path on the inventory beacon.

c. For any Oracle server which is both within the authorized subnet and listed in the `.ora` file, the inventory beacon checks for credentials in its Password Manager, and tests them (from the most closely matching to the most general) until either one gets a response (success), or there are no further applicable credentials (failure).

d. When successfully logged in, the FlexNet inventory agent running on the inventory beacon, uses the credentials to read the data necessary for Oracle introspection. It writes the data as an Oracle inventory file into its staging folder.

e. Within a minute of completion, the regular upload process starts moving this inventory file to the central application server (or, in a multi-server installation, the inventory server).

8. After the next inventory import and resulting consumption calculations, the Oracle inventory is available in the web interface for FlexNet Manager Suite; and the Oracle servers originally identified in the `TNSNames.ora` file are visible in the **All Discovered Devices** listing, displaying `Yes` in the **Oracle** column.

Prerequisites for the OEM Adapter

The OEM adapter must be installed on a computer with network access to Oracle Enterprise Manager, and requires read access to certain tables and data views there (the required permissions are listed in Grant Permissions to Account).

The OEM adapter requires that, on the computer where it will execute, the Oracle client version 12.1 is installed. Only the 32-bit version is supported: even for a Windows 64-bit computer, use 32-bit ODAC 12c Release 1 (12.1.0.1.0).

To take advantage of the information gathered by the OEM adapter, there are also the following requirements on the remainder of the system. Once the OEM adapter saves a `TNSNames.ora` file, the subsequent gathering of Oracle inventory requires that:

- The FlexNet inventory agent can access each Oracle system. This can be achieved either by installing the FlexNet inventory agent on the target Oracle server(s), or by remote execution (Zero-footprint inventory gathering, a term further explained in the *Gathering FlexNet Inventory* PDF, available through the title page of online help).
- You have FlexNet Manager for Datacenters, a separately licensed product within FlexNet Manager Suite.

Tip: *You can check whether your implementation has this product licensed in the web interface for FlexNet Manager Suite:*

1. *Navigate to the system menu (⚙ ▼ in the top right corner) >* ***FlexNet Manager Suite License****.*
2. *Check the list of* ***Subscribed and purchased products*** *on the right, looking for a card for FlexNet Manager for Datacenters. If the card is present, you have this product licensed.*

Components

The OEM adapter is supplied as an installer, called `OEMAdapter.exe`, which installs the following:

- OEM adapter executable and dependencies
- OEM adapter configuration file
- OEM adapter scheduled task, which can be created during installation if desired.

Download Adapter Tools Archive

The Adapter Tools archive includes content for many adapters, and is updated on the Flexera website from time to time.

Start this procedure using a web browser on a computer that has good network accessibility from all the machines needing installations for your adapter.

To download the adapter tools archive:

1. Download the `Adapter Tools for` FlexNet Manager Suite `2019 R2.zip` archive from the Flexera Customer Community knowledge base:

   a. Access https://flexeracommunity.force.com/customer/articles/en_US/INFO/Adapter-Tools-for-FlexNet-Manager-Suite.

   Tip: *Access requires your Customer Community user name and password. If you do not have one, use the link on the login page to request one.*

   b. Click the link `Adapter Tools for FlexNet Manager Suite`.

   A new browser tab may appear temporarily, and the download of `Adapter Tools for` FlexNet Manager Suite `2019 R2.zip` commences.

   c. In your browser dialog, choose to save the file, and if the browser allows it, direct the saved file to a convenient working location (such as `C:\temp` on a central, accessible server).

   If your browser saves the file to a default location (such as your `Downloads` folder), move or copy it to the appropriate working location when the download is finished.

2. Right-click the zip archive, and choose **Extract All...**.

The folders are now available for the range of adapters in the Adapter Tools archive.

2

Installing the Adapter, and More

This chapter covers the fairly simple process of installing the OEM adapter. However, once installed, the OEM adapter is only a small part of gathering Oracle database inventory. Therefore the remainder of this chapter assumes a fairly new implementation of FlexNet Manager Suite, and introduces the other kinds of set up necessary for a working system of Oracle introspection. Some of the latter may already be in place within your enterprise.

Installing the OEM adapter

The OEM adapter is normally installed on an inventory beacon that has high-speed network access to the Oracle server to which the adapter must connect. It is possible to install multiple instances of the OEM adapter on the same computer, each configured to access a different OEM repository. Each instance of the OEM adapter can access exactly one OEM repository.

To install the OEM adapter:

1. From the unzipped Adapter Tools archive, navigate into the `Oracle Enterprise Manager Adapter` folder.

   For details about downloading the archive, see Download Adapter Tools Archive.

2. In Windows Explorer, execute the `OEMAdapter.exe` installer from that folder.

   If this is the first installation of the OEM adapter on this computer, the welcome page appears, and you can click **Next**. If another instance of the OEM adapter is already installed on this computer, the following screen appears.

3. If this page appears, choose either of the following:

   a. For an additional installation, click **Install a new instance**, and click **Next**. Follow the remaining instructions below.

   b. To change one of the instances previously installed, select the instance from the list on this page, click **Maintain or upgrade an existing instance**, and click **Next**.

   The **Customer Information** page appears.

4. Identify the license owner, and the enterprise name, in the **User Name** and **Organization** fields respectively.

5. Provide credentials for the Windows scheduled task that triggers operation of the OEM adapter.

   Tip: *This account need not be the same as the one to access the OEM repository (identified in a later page).*

   Leave the fields blank to run the scheduled task as the local SYSTEM account on this computer.

   Note: *To change timing or frequency of the scheduled task, use the Microsoft Windows facilities as usual. They are not configurable through the installer.*

6. Click **Next >**.

   The **Destination Folder** page appears.

7. Optionally, click **Change...** to modify the supplied installation location.

> ***Note:*** *If you are installing multiple instances of the OEM adapter on this computer, you must modify the path so that each is installed in a separate folder. The InstallShield wizard offers the default folder `0` for the first instance. Increment this value to give a unique folder for each instance.*

> ***Important:*** *By default the OEM adapter saves the `.ora` file in the directory where it is executing (that is, the one you specify here). This is useful as a holding bay where the files may be manually inspected for initial testing/verification; but no automated processing of the `.ora` files occurs from this location. For automated operation after initial testing, you must reconfigure the file path and name as described in Configure Data Staging.*

8. When satisfied with the location, click **Next >**.

   The **Database Connection Information** page appears.

9. If necessary, confer with your Oracle DBA to complete details about the Oracle Database from which this instance of the OEM adapter collects inventory:

   a. In the **Host** field, identify the server where Oracle Enterprise Manager is installed.

   b. The **Port** is used by the OEM adapter to access Oracle Enterprise Manager.

c. **Service Name** identifies the Oracle service (for this database instance) that was defined when Oracle was installed.

d. **UserID** is the account name (with its related **Password**) accepted by Oracle Enterprise Manager for login by the OEM adapter.

   Suggestion: `FNMS-OEMadapter`.

When satisfied, click **Next >**, and the **Email Configuration** page appears.

10. To receive email alerts when the OEM adapter encounters any errors:

    a. Enter your email address as the **To Mail** value.

    b. In **From Mail**, enter the email address from which the error alerts should come.

    c. In the **SMTP** field, enter the fully qualified domain name (or IP address) of the email server.

    Tip: *Ensure that the **To Mail** and **From Mail** addresses are both recognized by this email server.*

    d. Click **Next >**.

    The **Ready to Install the Program** page appears.

11. When satisfied, click **Install**.

The installer writes the OEM adapter executable (`OEMAdapter.exe`) into your chosen folder, and records your other settings in a configuration file (`OEMAdapter.exe.config`) saved in the same folder. It also creates a scheduled task to run the OEM adapter. To allow for multiple instances of the OEM adapter on the same computer, the scheduled tasks are named based on the numbering of the installed instances. The default scheduled task for the initial installation on a computer looks like this:

You may update the scheduled task(s) as usual through the Microsoft interface.
Before using the OEM adapter in production, you must also do all of the following:

- Confirm that the account used to access Oracle Enterprise Manager (suggested name `FNMS-OEMadapter`) has adequate permissions to read from the OEM repository (see Grant Permissions to Account)
- Modify the location where the `.ora` file is saved for production use (see Configure Data Staging)
- Ensure that the appropriate subnets containing target Oracle servers are identified and assigned to the inventory beacon where the `TSNNames.ora` file is saved (see Assign Beacon to Subnet)
- Configure the collection of Oracle inventory in the web interface for FlexNet Manager Suite (see Configure Collection of Oracle Inventory)
- Set up accounts on each Oracle server with adequate permissions to gather inventory, with target machines being based on the contents of the .ora file created by the OEM adapter (see Inventory-Gathering Accounts on Oracle Servers)
- Register the same account(s) in the Password Manager for each relevant inventory beacon (see Save Inventory Account in Password Manager).

Grant Permissions to Account

The account used to access Oracle (suggested name `FNMS-OEMadapter`) must have adequate permissions to tables in the OEM repository. If you are setting up multiple instances of the OEM adapter, repeat the process for each adapter (you may be using a common account name for all of them to access each distinct OEM repository, or providing each with a unique account name). This task is normally completed by an Oracle DBA.

To provide appropriate permissions and validate operation:

1. To grant adequate read permission, do *either* of the following:
   a. Make the account (suggested name `FNMS-OEMadapter`) a member of the `EM_ALLVIEWER` role.
   b. Give the account read permissions on the following:
      - `DBA_TABLES`
      - `MGMT_TARGETS`

- `MGMT_TARGETS_LOAD_TIMES`
- `MGMT_TARGET_TYPES`
- `MGMT_TARGET_PROPERTIES`

2. When permissions have been set up, test as follows:
   a. Log in using the account name (suggestion: `FNMS-OEMadapter`) and password registered during the installation process.
   b. Execute the following query against the OEM repository:

      ```
      SELECT * from mgmt$Target
      ```

      Note the number of rows returned.
   c. Log out, and log in using a `DBroot` account (or other account known to have full permissions); and repeat the same query.

      The same number of rows should be returned by the root account with full permissions, and by the account for the OEM adapter with more limited and specific permissions.
3. Repeat the process and testing for any further instances of the OEM adapter accessing other distinct instances of OEM repository.

Other Setup Activities

The OEM adapter is relatively straight-forward, and, as triggered by the Windows scheduled task set up during installation, regularly collects data from Oracle Enterprise Manager and saves the data as a `TNSNames.ora` file in a special location on the inventory beacon.

However, by itself, when the goal is to gather inventory from Oracle Database servers, the `TNSNames.ora` file is just the first step. Many other 'cogs in the machine' need to do their part for the `.ora` file to be beneficial:

- You need a defined set of sites and subnets, most easily obtained by Active Directory import through an inventory beacon.
- The subnet(s) containing your Oracle servers of interest must be assigned to the inventory beacon that collects the `TNSNames.ora` file, and which subsequently collects the Oracle inventory (see Assign Beacon to Subnet).
- An inventory rule must be defined that associates the target subnet with an action including inventory collection using the `TNSNames.ora` file, and schedules its execution (see Configure Collection of Oracle Inventory).
- Quite separately from the account that queries the OEM repository to create the `TNSNames.ora` file, one or more separate accounts must be defined that connect to the Oracle Database servers and collect the inventory data. There are two parts to configuring these accounts:
  - The accounts must be created and given adequate permissions on each Oracle Database server (there is a utility available to assist with this, as described in Inventory-Gathering Accounts on Oracle Servers)
  - The same accounts must be registered in the Password Store on the inventory beacon (see Save Inventory Account in Password Manager).

Strictly speaking, none of these are part of the OEM adapter; but all are integral to the process of Oracle introspection, and therefore are at least summarized in the following sections.

Inventory-Gathering Accounts on Oracle Servers

The inventory beacon collects inventory from Oracle servers using accounts that must be registered at both ends.

To set up inventory-gathering accounts on an Oracle server:

1. Using your credentials supplied as part of order confirmation/fulfilment, log in to the Flexera knowledge base available through the Support pages of the Flexera website, and access the following knowledge article:

   ```
   https://flexeracommunity.force.com/customer/articles/en_US/INFO/Q200934
   ```

2. Scroll to the bottom of the article, and click the link `CreateOracleAuditUserQ200934.sql`.

   This downloads a SQL script that an Oracle DBA can review. (Ignore the text content which is for earlier products.)

3. In a flat text editor, modify lines 8 and 9 of the script, replacing the default user name and password with credentials of your choosing; and save the modified script.

   To minimize configuration and maintenance effort, the same credentials can be implemented on each Oracle Database server. Keep a note of these credentials, as you must also record them in the Password Manager on the inventory beacon.

4. An Oracle DBA can run the script on each target Oracle Database server.

   This (re)creates the user name on each run, and grants read access to a numbers of tables and views (shown below) that are required for Oracle Database introspection.

The downloaded SQL script gives the account read-only access to the following tables and views:

- `applsys.fnd_app_servers`
- `applsys.fnd_nodes`
- `applsys.fnd_product_installations`
- `applsys.fnd_application_tl`
- `applsys.fnd_user`
- `applsys.fnd_responsibility`
- `apps.fnd_user_resp_groups`
- `SYS.DBA_USERS`
- `SYS.V_$PARAMETER`
- `SYS.V_$INSTANCE`
- `SYS.V_$DATABASE`

- SYS.V_$OPTION
- SYS.DBA_FEATURE_USAGE_STATISTICS
- SYS.DBA_ENCRYPTED_COLUMNS
- SYS.DBA_TABLESPACES
- ODM.ODM_MINING_MODEL
- ODM.ODM_RECORD
- DMSYS.DM$OBJECT
- DMSYS.DM$MODEL
- DMSYS.DM$P_MODEL
- DVSYS.DBA_DV_REALM
- LBACSYS.LBAC$POLT
- OLAPSYS.DBA$OLAP_CUBES
- SYS.DBA_AWS
- SYS.DBA_SEGMENTS
- SYS.DBA_CUBES
- SYS.GV_$INSTANCE
- SYS.GV_$PARAMETER
- MDSYS.ALL_SDO_GEOM_METADATA
- SYS.V_$SESSION
- SYSMAN.MGMT_LICENSE_DEFINITIONS
- SYSMAN.MGMT_ADMIN_LICENSES
- SYSMAN.MGMT_LICENSES
- SYS.DUAL
- SYSMAN.MGMT_LICENSE_CONFIRMATION
- SYSMAN.MGMT_TARGETS
- SYSMAN.MGMT_$TARGET
- SYSMAN.MGMT_VERSIONS
- SYSMAN.MGMT_INV_COMPONENT
- SYSMAN.MGMT_FU_REGISTRATIONS

- SYSMAN.MGMT_FU_STATISTICS
- SYSMAN.MGMT_FU_LICENSE_MAP
- SYS.DBA_REGISTRY
- SYS.V_$LICENSE
- SYS.DBA_TABLES
- CONTENT.ODM_DOCUMENT
- SYS.V_$VERSION
- SYS.USER_ROLE_PRIVS
- SYS.USER_SYS_PRIVS
- SYS.ROLE_SYS_PRIVS
- MDSYS.SDO_GEOM_METADATA_TABLE
- SYS.DBA_INDEXES
- SYS.DBA_LOBS
- SYS.DBA_OBJECTS
- SYS.DBA_RECYCLEBIN
- SYS.DBA_MINING_MODELS
- SYS.REGISTRY$HISTORY
- SYS.DBA_TAB_PARTITIONS
- SYS.DBA_TAB_SUBPARTITIONS
- SYS.DBA_LOB_PARTITIONS
- SYS.DBA_LOB_SUBPARTITIONS
- SYS.V_$ARCHIVE_DEST_STATUS
- SYS.DBA_SQL_PROFILES
- SYS.DBA_ADVISOR_TASKS
- SYS.DBA_SQLSET
- SYS.DBA_SQLSET_REFERENCES
- SYS.DBA_FLASHBACK_ARCHIVE
- SYS.DBA_FLASHBACK_ARCHIVE_TS
- SYS.DBA_FLASHBACK_ARCHIVE_TABLES

- `SYS.V_$BLOCK_CHANGE_TRACKING`
- `SYS.V_$CONTAINERS`

Save Inventory Account in Password Manager

The accounts that collect Oracle inventory must exist in the Password Manager.

This process must be completed on the inventory beacon.

To register an account in the Password Manager on an inventory beacon:

1. Log into the inventory beacon, using an account that has administrator privileges.
2. Start the FlexNet Beacon software from the Windows Start menu.
3. In the navigation bar on the left, select the **Password management** page, and click **Launch Password Manager**.

   The separate interface for the Password Manager opens.
4. In the **Current credentials** group, click **New**.

   The controls in the **Editor** group are activated. When you are updating an existing entry, the controls are populated with the currently saved values.
5. Provide a **Logical name** to identify this set of credentials.

   Logical naming allows you to have one account name, but with different passwords on different servers. For more, see the online help.
6. For **Account type**, choose `Account on Oracle database`.
7. Complete the **User** (account name) and **Password**.
8. Click **View/Edit...**.

   The **Password Store: Password Filter** dialog displays.
9. Use the **Oracle service names** filter to create a comma-separated list of the Oracle services for which this account/password pair should be used.

   The services names are visible in your `.ora` file. Filtering the account/password pair to apply only to specified Oracle services provides maximum efficiency for login and introspection.
10. Click **Apply** to close the dialog, and continue to save the entry in the Password Manager.

    Repeat the process as required until all your Oracle machines are covered by one or more entries in the Password Manager. Thereafter you may exit the FlexNet Beacon interface.

Assign Beacon to Subnet

Before rules can take effect, inventory beacons must know their subnets.

It is a requirement for an operational system that your subnets are assigned to appropriate inventory beacons. This summary covers only making adjustment for the collection of Oracle inventory.

To assign subsets to an inventory beacon:

1. In the web interface for FlexNet Manager Suite, navigate to **Discovery & Inventory > Subnets** (in the **Network** group).

   This page is populated with the sites and subnets in your enterprise after you have completed an import from Active Directory (for details, consult *FlexNet Manager Suite Help > Inventory Beacons > Active Directory Page > Importing from Active Directory*).

2. Expand the appropriate site(s), using the **+** expander icon, until you can see the subnet that includes your Oracle Database servers.

   Tip: *If the subnet does not yet appear in the listing, you can add its details manually. In the row for the appropriate site, click the* **+** *sign on the right-hand end, and enter the subnet IP address in the new row that appears.*

3. If the **Beacon name** column shows `Unspecified`, click the editing (pencil) icon at the right-hand end of this row.

   The row becomes editable. Beware of accidentally over-writing the IP address, which initially has focus.

4. In the **Beacon name** column, use the drop-down list to choose the appropriate inventory beacon from the list of those registered so far; and click the blue disk icon on the right to save your change.

Your chosen inventory beacon is now authorized to work in the subnet that contains your Oracle Database servers. Now continue and create a rule that dictates what should happen within that subnet.

Configure Collection of Oracle Inventory

Rules must be established that allow Oracle inventory collection, including the use of `.ora` files.

Inventory rules are created on the Discovery and Inventory Rules page of the web interface for FlexNet Manager Suite. Each rule has three parts:

- A *target* that identifies the machines on which the rule is to be exercised
- *Actions* that are the be performed on those target machines
- The *schedule* on which the rule is to be applied.

When an action includes permission to use a `.ora` file, the relevant inventory beacon uses the locally-available `.ora` file to 'filter' the related target and identify the Oracle Database servers from which inventory should be collected. For example, if you target an IP range that covers your server room, and include the action setting to apply the `.ora` file to this range, then Oracle inventory is collected only from the matching machines listed in the `.ora` file.

Once a discovery and inventory collection has been completed, the individual Oracle Database servers are identified in the list of discovered devices. Should you wish to change to tighter targeting rules, you can use this information to redefine the target until (if you wish) it identifies exactly the Oracle Database servers and no others. Keep in mind that if you use this approach, you will need to adjust the target each time you vary the list of your Oracle Database servers. To reduce this manual maintenance, keep a target that specifies an appropriate IP range (or ranges), and applies the `.ora` file to this to identify the individual Oracle Database servers from which to collect inventory.

To configure Oracle inventory collection:

1. In FlexNet Manager Suite, navigate to **Discovery & Inventory > Discovery and Inventory Rules** (in the **Discovery** group).
2. If you do not already have a target to reach the Oracle servers from which you want to gather inventory:
   a. On the left side, select the **Targets** tab.

      Fastpath: *In the hints area across the top of the page, click* ***Create targets****. These hints can guide you through the process.*

   b. On the right side, click **Create a target**.
   c. The page appearance changes, allowing you to define a target.
   d. Complete the details requested, with particular attention on **Define machines to target**.

      Notice that:

      - After you complete each definition for this control, a **+** icon is displayed that allows you to add more to your definition of target machines. Use these lines to define a target sufficiently broad to capture your Oracle Database servers.
      - If you do not want the FlexNet inventory agent to be installed on these Oracle servers, be sure to select **Do not allow these targets to be adopted**.
      - You should likely also select **Do not allow application usage tracking on these targets**, since you may not want to collect a large quantity of file evidence from these servers.

   e. Click **Save**.
3. If you already have an action for Oracle discovery, you can check its settings; or create a new one:
   a. Click the **Actions** tab (or in the hints section, click **Create actions**).
   b. If you wish to collect hardware inventory for these servers (perhaps to assist with licensing calculations for your Oracle Database license), expand the General accordion and select **Gather hardware and software inventory**.

      Tip: *You may want to clear the check box for* ***Discover devices****, if you are limiting this action specifically to inventory collection for known Oracle Database servers identified in your* `.ora` *file.*

   c. Expand the **Oracle database** accordion.
   d. If you are confident that every Oracle Database is identified in your `.ora` file, you may clear the check box for **Discover Oracle databases**. Alternatively, if you may have rogue servers, leave this check box selected, and the network within your declared target will be probed for other Oracle servers.
   e. Select the check box for **Also gather Oracle database inventory**.

      Additional controls, if not already visible, are exposed.

   f. Ensure that the **Port scan** check box is selected, and if necessary use the **+** icon to add additional ports to the list until every port listed in your `.ora` file is included.

g. Ensure that the **SNMP scan** check box is selected.

h. It is critically important that you select the **TNS names file** check box.

   This setting authorizes the relevant inventory beacons to apply any `.ora` files found in the 'magic path' to the target used for this action (in the rule soon to be completed). This is the mechanism that most efficiently restricts probing to the relevant servers.

i. Adjust other settings in other parts of the accordion to suit your environment, and click **Save**.

   The action is stored, ready for inclusion in a rule.

4. Select the **Rules** tab (or in the hints area, click the third **Create rules** step).

   a. Do either of the following:

      - If you have an existing rule to review or modify, click the edit (pencil) icon on its right-hand end.
      - Click **Create a rule** (upper right) to define a new rule (as described in the following steps).

      A rule builder work area appears above the list of existing rules.

   b. Return to the **Actions** tab (for example, using the link in the rule builder), and on the row for your edited Oracle action, click **Add to rule builder**.

      The name of your action appears in the rule builder.

   c. Return to the **Targets** tab (for example, using the link in the rule builder), and on the row for your edited target, click **Add to rule builder**.

5. On the right side of the rule builder, click **Schedule**.

   The rule builder changes to display controls for scheduling:

   a. Choose a value from the **Frequency** drop-down list.

      This control sets the style of scheduling. For example, the `Daily` option lets you make further choices about a pattern of days, and does not enforce inventory collection every day.

Option	Notes
`Once`	A single-shot trigger for inventory collection the next time the declared time window occurs (you cannot nominate a future date). For example, if it is 4pm when you set a `Once` schedule for 8am, commencing within 4 hours, inventory collection normally occurs next morning. Compare with `As soon as possible`. Keep in mind the propagation delays for your changed instructions, as described there.

Option	Notes
`Daily`	An additional drop-down list, **Every**, appears so that you can choose the pattern of days you want, extending the time intervals between inventory collection in multiples of 24 hours. For example, you may wish to trigger inventory collection every second day. The starting time is within the **Commence within** time window following the next occurrence of your **Start time** setting after the changed schedule is distributed (and received by each installed FlexNet inventory agent). For example: • **Start time** is set to `8:00 am`. • **Commence within** is set to `2 hours`. • The schedule is saved at noon today, and by default the inventory beacon policy is distributed every 15 minutes. • By default, each FlexNet inventory agent chooses a random time within a one-hour window from 5am local time to request any policy (and schedule) changes. Most devices therefore start using the changed schedule tomorrow morning. In short, the setting applies individually to each installed FlexNet inventory agent, and does not synchronize the activities of all agents across your enterprise to identical days.
`Weekly`	New check boxes appear that allow you to choose specific days within the weekly cycle when inventory should be collected. For example, you may want collection on Sunday and Wednesday every week. Select (check) the boxes for the days you prefer.
`Monthly`	New controls appear that allow you to specify a pattern within the month: • Choose the option for **On day** to nominate a particular day within the month (such as the third Saturday). Make your choices from the two drop-down lists adjacent. Notice that the option `Last` chooses the fifth occurrence in months long enough to have one, and otherwise takes the fourth occurrence. • Choose the option for **On date** to nominate the calendar date within the month.

Option	Notes
`As soon as possible`	This is a single shot trigger which causes each FlexNet inventory agent to randomize an inventory collection within the time window starting when it receives this setting and lasting for the interval you specify in the **Commence within** controls. ***Tip:*** *Any change you save to these settings is first collected by the inventory beacons on the schedule specified by the* ***Beacon settings*** *(further down this web page), by default checked every 15 minutes. Each inventory beacon then prepares new instructions for any installed FlexNet inventory agents that request an update to device policy (the operational FlexNet inventory agent checks for updated policy once per day, randomized across a one-hour window in the early morning; but newly installed agents seeking their first policy make a request once every 12 hours on UNIX-like platforms, or once every system restart on Windows). Because of these propagation periods,* `As soon as possible` *typically means starting early tomorrow morning.*

6. In the rule builder, click **Save as**, give the rule a name you will recognize later in lists, and click **Save**.

If you accepted the default `Enabled` setting, the rule is ready to run on the schedule you have established. (Remember to allow around 15 minutes for the new rule to be distributed to your inventory beacons.) When the rule is executed on an inventory beacon, if a `.ora` file exists in the 'magic path', the systems that lie both within the inventory beacon's assigned subnet(s) and within the `.ora` file are targeted for Oracle inventory collection.

3

Modifying the Adapter

This chapter covers changes you can make to the operations of the OEM adapter by modifying its configuration file. Read the first topic for general guidance on the editing process, and choose the detailed subtopics based on what changes you need.

Reconfiguring the OEM Adapter

During installation, you recorded your preferred settings for the OEM adapter, and no further work is required in the installation process. You can however choose to modify the configuration of an instance of the OEM adapter at a later stage.

To reconfigure the OEM adapter:

1. In Windows Explorer, navigate to the correct folder for the appropriate instance of the OEM adapter.

   Keep in mind that there may be several instances installed on a computer, accessing distinct instances of Oracle Enterprise Manager.

2. In your preferred plain text (or XML) editor, open the OEM adapter's configuration file, called `OEMAdapter.exe.config`.

   It is strongly recommended that you make a backup copy of the original configuration file, so that you can easily revert your changes if there are problems.

3. Use the following subtopics to guide your changes to the configuration file.
4. When you have finished, save the updated file.
5. Either wait until the next scheduled run of the OEM adapter, or use the Windows scheduled task interface to trigger an immediate test run to confirm that your changes work as expected.

Updating Connection Details

You can change all details and credentials for connection to the Oracle Enterprise Manager instance.

When the OEM adapter is first run, it encrypts the connection details used to access the Oracle Enterprise Manager

implementation. Therefore, the process to edit the connection details is different, based on whether the OEM adapter has been run since the connection details were last recorded. You can tell whether this is so by looking in the configuration file.

The details are all stored in the `<connectionStrings>` element of the configuration file.

1. If the OEM adapter has been run since the details were recorded in the configuration file, the encrypted `<connectionStrings>` element appears similar to this:

```
<connectionStrings
configProtectionProvider="DataProtectionConfigurationProvider">
    <EncryptedData>
      <CipherData>
        <CipherValue>AQAAANCMnd8BFdERjHoAwE/Cl+sBAAAAeNhcGMAVK0uVGNTqOg/
WbQQAAAACAA

AAAAADZgAAwAAAABAAAACTO9kpn6BptpLvsXExg1UBAAAAAASAAACgAAAAEAAAAELCyiwz5A

XZw9xZXfEPiAAAAgAAPJQYe+G9AfScFMJTYgA0NDbAgZdRA9nB91DN42A1xjeCskUs9+KNjV
           U1PSFRV4ujta40evf3IOZy5odyHsIrJRCKOOGdhDb1wh4ISEkpJk/
QDna6LeCbbtXXsQK2Lo
           AHQc/plz77UkQZxnxkL5ElPIGHl6AojEXT2F5NGjElJX6GXbUXQDkNnDfi2o6XI/
CDbX8gCu
           MonY1cTLYGe6+AQPpDgcY3rA02ZFOs7/Zb0cKOw7IoZdB6H8OIvrClSzqNkVBd3YfLhP/
KOr
           kQFP8orqj54BJW74E1v3VUZnte1ESgLA5MYb/
F9Ah3M5xi2Q6ITXOQmVRGESrivsdqr6nyGz
           5APx2yVBuEcoVhpOMYURbEbBSW+6/
aydg8nY1DrcMzkPlXiZ0CQs4yZYHSWt3+bFEN30xh6X

KFH7Tpc8e5y9Tq1Bhwk+Kw6AFfZhcdewApJ4ZkGAr4ixE6gNqWXnomk2puKNFImhJfqLLIuQ

x9T00Uu2bjJ9Y+dUlrcuov12hQXOCh9nqOdmeIQHPXgw8//eHYOzwy2TgMcq2M2LxXRbQqCT

eWtlP1ucJVyct9gnJlk2L3FSBNgMu1C9mncR7MsGDBWoQMXnwC5+Y6P1GT45sPOedBQvIQIT

j7MCuzfgDD1R9c7w1gejTGmrl3Kmf33tGPMRYPV7CPNET3DUV9AGQUAAAAaIEkjyqfExrbei
           YJvG2usqYO3Nk=</CipherValue>
      </CipherData>
    </EncryptedData>
  </connectionStrings>
```

   In this case, delete the entire element and replace it with a plain text version as shown below.

2. If the OEM adapter has not been run since the configuration file was changed, it has a plain text form similar to this (line wrapping has been added here for legibility). Replace the italicized placeholders with your own values, as described below. Keep the `connectionString` attribute all on one line.

```
<connectionStrings>
    <add name="OEMConn"
```

```
        connectionString="Data Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)
                         (HOST=HostIPAddress)
                         (PORT=PortNumber))
                         (CONNECT_DATA=(SERVICE_NAME=ServiceName)));
        User Id=AccountName;Password=UnencryptedPassword;"></add>
    </connectionStrings>
```

where the following details should be confirmed with your Oracle administrator:

- *HostIPAddress* is the IP address of the Oracle server hosting Oracle Enterprise Manager
- *PortNumber* is the port that the OEM adapter should use to query Oracle Enterprise Manager
- *ServiceName* is the service name established when Oracle was installed on the Oracle server
- *AccountName* is the account (user name) that the Oracle administrator has established for accessing Oracle Enterprise Manager
- *UnencryptedPassword* is a plain text rendition of the password for the same account (remembering that all the connection details are encrypted together as soon as the OEM adapter runs next).

3. Save the amended configuration file.
4. Use the Windows schedule task interface to run a test of the OEM adapter to confirm that the connections details were correctly entered.

Configure Data Staging

Modify the name and location of the file of Oracle connection information.

The OEM adapter collects information about connections to Oracle systems from which you need to gather inventory information. This file is in a standard Oracle format, used for their `TNSNames.ora` file. It is saved by default where the OEM adapter is executing; but the default location for the OEM adapter does not support automated processing of the `TNSNames.ora` file when the inventory beacon applies rules for gathering Oracle inventory. For the OEM adapter to function seamlessly, you must customize the location as you edit the `OEMAdapter.exe.config` file.

Another reason to customize the file name is if you have multiple instances of the OEM adapter running from the same computer (or if anyone manually adds `TNSNames.ora` files to the processing directory). Each instance must write to a unique file name, so that one output does not over-write the other.

These settings live in an `add` element with the `key` attribute of `ConnectionInfoFile`.

To configure data staging:

1. In the `OEMAdapter.exe.config` file, locate the appropriate element. Its default values are similar to the following:

```
<add key="ConnectionInfoFile"
     value="C:\Program Files\Flexera\Oracle Enterprise Manager Adapter\
TNSNames.ora"></add>
```

This reflects the default location of the OEM adapter, and must be modified.

2. Replace the `value` string with the new file path and name.

   Important: *The file name* ***must*** *use the extension* `.ora`*.*

   You may use a mapped drive on the local computer to specify a network path. This example shows the file in the recommended 'magic path' on the inventory beacon where the `.ora` file is automatically processed to 'filter' the target supplied from the central application server. Also notice the customized file name to avoid naïve overwriting of the `TNSNames.ora` file name by other copies saved here:

   ```
   <add key="ConnectionInfoFile"
        value="C:\ProgramData\Flexera Software\Repository\TNSNames\
   TNSNames-01.ora"></add>
   ```

3. Save the amended configuration file.

4. Use the Windows schedule task interface to run a test of the OEM adapter to confirm that the staging file is saved according to your revised specifications. Check the location (recommended: `C:\ProgramData\Flexera Software\Repository\TNSNames\`) for the presence of a saved file immediately after the test run (if you wait too long, the resulting file may be automatically uploaded and removed from this staging location).

Managing Email Alerts

You can turn the alerts off, changes their addresses, or switch email servers.

The OEM adapter can send email alerts any time that it encounters an error. Manage the emails with the following XML elements in the `OEMAdapter.exe.config` file.

1. To stop email alerts entirely comment out the *reference* to the `SmtpAppender` like this:

   ```
   <appender-ref ref="RollingFileAppender"></appender-ref>
   <!-- <appender-ref ref="SmtpAppender"></appender-ref> -->
   ```

   (The character sequence `<!--` starts an XML comment, and the sequence `-->` closes the comment.)

   Tip: *Using this technique is preferable to deleting this line entirely, as this can be easily reversed if you decide to reinstate email alerts in future. Notice that, depending on the configuration you saved, there may be other elements between the* `<appender-ref>` *tags.*

2. To reconfigure the email alerts, locate the `<appender>` element, and modify three of its child elements by changing the example values shown in italics here:

   ```
   <appender name="SmtpAppender" type="log4net.Appender.SmtpAppender">
      <to value="toaddress@somedomain.com"></to>
      <from value="fromaddress@somedomain.com"></from>
      <subject value="OEM Adapter Error"></subject>
      <smtpHost value="smtp.somedomain.com"></smtpHost>
   </appender>
   ```

3. Save the amended configuration file.

Configure Logging

Change the path and file name for logging by the OEM adapter.

The log file name and location cannot be set during installation of the OEM adapter, but after installation, they can be modified as follows:

To configure logging:

1. In the `OEMAdapter.exe.config` file, locate the `<appender>` element called `RollingFileAppender`.
2. Edit the child `<file>` element by replacing the drive, path and file name shown as placeholders here:

```
<appender name="RollingFileAppender"
          type="log4net.Appender.RollingFileAppender">
   <file value="drive:\path\path\logfilename.txt"></file>
</appender?
```

3. Save the amended configuration file.

After the next run of the OEM adapter, inspect the log in your new location.

XI

Salesforce Subscription Management

The benefits of Salesforce subscriptions come at a significant cost for large organizations and licensing is complex. FlexNet Manager Suite's provides you with ways to better stay in charge of your Salesforce licensing and consumption. FlexNet Manager Suite provides an inventory adapter so that you can import Salesforce license allocations and usage data and then leverage FlexNet Manager Suite's tools to help you manage Salesforce licensing and consumption costs. With FlexNet Manager Suite's integration with Salesforce, you can easily see unused Salesforce subscriptions, thereby helping you strategize how to reduce costs at subscription renewal. FlexNet Manager Suite:

- Imports Salesforce license information and creates one SaaS User license for each instance of Salesforce that is connected. In FlexNet Manager Suite, under the SaaS User license, a series of applications represents the Salesforce users in your organization.
- Shows you consumption information in the **Consumption** tab of the license properties for each user in your organization that is accessing Salesforce.
- Shows unused Salesforce subscriptions. To determine unused Salesforce subscriptions, you can specify a number of days in a **SaaS Usage Summary** report and drill through to users to view which users have and have not accessed Salesforce over that time period.
- Supports multiple Salesforce instances.
- Provides a list of accessing users for each Salesforce instance.

Note: *You must be connected to Salesforce in order for FlexNet Manager Suite to provide Salesforce information.*

1

Salesforce License Considerations

You can use FlexNet Manager Suite to review your Salesforce subscriptions. FlexNet Manager Suite provides a way for you to:

- Review Salesforce license consumptions to help determine how many Salesforce licenses you need to purchase, which in turn helps you negotiate license agreements.
- Ensure that business units are not under-subscribed.

Note: *The first time you import your organization's license information from Salesforce, FlexNet Manager Suite creates one SaaS User license for each instance of Salesforce that is connected. If license information for a new user is imported after the initial import, that information is imported to the original SaaS User license in FlexNet Manager Suite. Each user imported from Salesforce is mapped to the appropriate accessing user record. These users consume against the Salesforce multi-product license created for the Salesforce tenant.*

2

Viewing Salesforce License Information with FlexNet Manager Suite

The following provides high-level steps for viewing Salesforce license information using FlexNet Manager Suite.

To view Salesforce license information:

1. Ensure that you have created and configured a connector for each tenant of the Salesforce Online Service. For more information, see Managing Connections to Salesforce.com.
2. During the next discovery and inventory collection, FlexNet Manager Suite imports the following objects:
   - **Users**: Each imported user from Salesforce.com is mapped to the appropriate accessing user record. If an imported user cannot be mapped to any accessing user record, an accessing user record is created with a dummy inventory device assigned to this user.
   - **Licenses**: A `SaaS User` license is created for each Salesforce tenant (typically one tenant per enterprise, although mergers and acquisitions may mean that your enterprise has multiple Salesforce tenants). Note the following:
     - This license is derived from a template provided in the downloadable libraries, and selected based on the plan identifier.
     - Multiple applications are linked to this license that represent feature and user licenses within Salesforce.
     - The name of the Salesforce tenant is included in the name of each of these license and applications. This helps to distinguish licenses when you have multiple Salesforce tenants.
3. Review the **All Users** and **All licenses** pages to ensure that the required users and licenses have been created.
4. After license reconciliation has been completed, the **Product Summary** page should show the compliance status for Salesforce.
5. Navigate to **License Compliance > All Licenses** and filter the records to view Salesforce licenses. You can check the **Consumed** and **Used** columns to know how many subscriptions are being used.

6. You can also view the SaaS Usage report for insights on Salesforce license consumption.

3

Connecting to the Salesforce Online Service

To generate the compliance position for Salesforce licenses, FlexNet Manager Suite needs information about subscription and usage of Salesforce licenses. To get the subscription and usage information from Salesforce.com, the FlexNet Beacon should be configured with an inventory PowerShell connection of the **Source Type** of `Salesforce` for each tenant of Salesforce. (Typically, each enterprise is a single Salesforce tenant; but your corporate history, including mergers and acquisitions, may mean that your enterprise includes multiple Salesforce tenants.)

When created and configured with the Salesforce connection, the inventory beacon imports the licenses, users, and usage information from the Salesforce.com account and uploads it to FlexNet Manager Suite according to the defined schedule. according to the defined schedule. For details on how to create a connector to Salesforce, see Managing Connections to Salesforce.com.

Managing Connections to Salesforce.com

FlexNet Manager Suite 2019 R2 (On-Premises Edition)

Use the following procedure to create a connection to Salesforce.com on the FlexNet Beacon. FlexNet Manager Suite provides an inventory adapter to import Salesforce.com license allocations and usage data. Operators can easily see unused Salesforce.com subscriptions which, in turn, allows you to manage costs at subscription renewal. A connection is required for each organization of Salesforce.com. The inventory beacon requires this connection to import all Salesforce.com licenses available to the enterprise, show consumption for each allocated license, show unused subscriptions, and support multiple Salesforce.com instances.

You must make sure that the Salesforce tenant user has the required System Administrator privilege for this operation. Also make sure that the following prerequisites are met on each inventory beacon that needs to download data from Salesforce (noting that the order of installation of prerequisite software may be significant). These requirements should have been met when the inventory beacon was installed:

- PowerShell 5.1 or later is running on Windows Server 2008 R2 SP1 or later, or Windows 7 SP1 or later; with the PowerShell execution policy set to `RemoteSigned`

Tip: *The PowerShell execution policy can be set by running PowerShell with administrator rights and executing*

the following command:

```
Set-ExecutionPolicy RemoteSigned
```

- A browser is installed with JavaScript enabled on the machine the inventory beacon software is installed on.
- A Javascript-enabled browser installed on the machine the inventory beacon software is installed on
- The inventory beacon server supports Transport Layer Security (TLS) 1.2 in order for the connection to Salesforce to work.

To create a connection to Salesforce.com:

1. Login to Salesforce and create a connected app. (A connected app is required for a connection to Salesforce.com on the FlexNet Beacon to work.) For information about how to create a connected app in Salesforce, see https://help.salesforce.com/articleView?id=connected_app_create.htm. Note the following:

   When creating the connected app, ensure that you select **Enable OAuth Settings**, select the **Enable for Device Flow** check box and select all **Available OAuth Scopes**. The connected app in Salesforce uses standard SAML and OAuth protocols to authenticate, provide single sign-on, and provide tokens for use with Salesforce APIs. In addition to standard OAuth capabilities, connected apps allow Salesforce admins to set various security policies and have explicit control over who can use the corresponding apps. After creating the connected app in Salesforce, you can view the app details in order to copy and paste the **Consumer Key** and **Consumer Secret** values to the corresponding fields in the FlexNet Beacon PowerShell Source Connection dialog in **Step 8**.

2. Login to Salesforce and access the connected app to view details.

3. In the FlexNet Beacon interface, ensure that you have your preferred schedule for imports from Salesforce set on the appropriate inventory beacon:

   a. Log into the inventory beacon interface as an administrator (for example, in the Windows Start menu, search for **FlexNet Beacon**, right-click it, and select **Run as administrator**).

   Tip: *Remember that you must run the inventory beacon software with administrator privileges.*

   b. From the **Data collection** group in the navigation bar, choose **Scheduling**.

   c. If there is not already a suitable schedule in the list, click **New...** and complete the details (see the online help for that page for more information). Otherwise, identify the schedule you will use.

4. Select the **Inventory Systems** page (in the same navigation group).

5. Choose either of the following:

   - To change the settings for a previously-defined connection, select that connection from the list, and click **Edit...**.
   - To create a new connection, click the down arrow on the right of the **New** split button, and choose **Powershell**.

6. Complete (or modify) the values for the following required fields:

   - **Connection Name**: Enter a name for the inventory connection. The name may contain alphanumeric characters, underscores or spaces, but must start with either a letter or a number. When the data import

through this connection is executed, the data import task name is same as the connection name.

- **Source Type**: Select `Salesforce` from this list.

7. Optionally, if your enterprise uses a proxy server to enable Internet access, complete (or modify) the values in the **Proxy Settings** section of the dialog box in order to configure the proxy server connection.

   - **Use Proxy**: Select this check box if your enterprise uses a proxy server to enable Internet access. Complete the additional fields in the **Proxy Settings** section, as needed. If the **Use Proxy** check box is not selected, the remaining fields in the **Proxy Settings** section are disabled.
   - **Proxy Server**: Enter the address of the proxy server using HTTP, HTTPS, or an IP address. Use the format `https://ProxyServerURL:PortNumber`, `http://ProxyServerURL:PortNumber`, or `IPAddress:PortNumber)`. This field is enabled when the **Use Proxy** check box is selected.
   - **Username** and **Password**: If your enterprise is using an authenticated proxy, specify the user name and password of an account that has credentials to access the proxy server that is specified in the **Proxy Server** field. These fields are enabled when the **Use Proxy** check box is selected.

8. Complete (or modify) the values in the **Salesforce** section of the dialog box. All of the following fields require a value:

   Note: *If you have multiple organizations to Salesforce (for example, separate organizations for different corporate units or locations), you need to create a separate connector for each organization using its own credentials.*

   - **Salesforce URL**: Enter the address of the Salesforce URL to be used for generating a new token.
   - **Consumer Key**: Copy this value from the `Consumer Key` field in the Salesforce connected app.
   - **Consumer Secret**: Copy this value from the `Consumer Secret` field in the Salesforce connected app. In the Salesforce connected app, click **Click to reveal** to view the value.
   - **Refresh Token**: Click the **Generate** button to generate a refresh token that will be used to authenticate the connection to Salesforce.

9. When you click **Generate** to the right of the **Refresh Token** field, a Web browser is launched in Salesforce.com with an 8-digit code automatically populated in the **Code** field. In the Salesforce.com screen, do the following:

   a. Click **Connect**. A Salesforce login page displays. If you are already logged into Salesforce, skip to step 8e.
   b. Enter your Salesforce username. If there are multiple Salesforce accounts, select your username from the **Saved Username** list *or* click **Log In with a Different Username**. A **Password** field appears.
   c. In the **Password** field, enter your Salesforce password.
   d. Click **Login**. An **Allow Access** page appears.
   e. Click **Allow** to allow Salesforce to access the FlexNet Beacon to have the refresh token sent back to PowerShell Source Connection dialog. A message appears to notify you that the connection is successful. Click **Continue** or close the browser.

      Note: *The following table provides help with potential issues that you may encounter when attempting to connect to Salesforce.*

Table 1: Troubleshooting Salesforce Connection Errors

Error	Description
`Mandatory parameter(s) missing`	One or more of the fields in the FlexNet Beacon PowerShell Source Connection dialog is missing a value. Ensure that all mandatory fields contain values before clicking **Generate** to the right of the **Token** field.
`The remote name could not be resolved: 'login.salesforce.com'`	The most common cause is that there is no Internet connection available on the machine the inventory beacon software is installed on. There may be other reasons such as Salesforce.com being blocked or is currently down.
`invalid_client_id`	The value that you entered for the **Consumer Key** field does not match the **Consumer Key** value from the Salesforce connected app. Copy the correct **Consumer Key** value from the Salesforce connected app and paste it into **Consumer Key** field in the FlexNet Beacon PowerShell Source Connection dialog.
`Timeout`	When the Web browser (that is invoked after clicking **Generate** to the right of the **Token** field) is not responding in a timely fashion, a `Timeout` error occurs. Some of the most common scenarios that will trigger this error are: • The Internet browser that was invoked when you clicked **Generate** was subsequently closed inadvertently or prematurely. • A login error was encountered that may be the result of incorrect login credentials. • You may have waited too long before attempting to login to Salesforce.
`invalid_request`	You have clicked **Deny** instead of clicking **Allow** in Step 5e to access the FlexNet Beacon to have the refresh token sent back to PowerShell Source Connection dialog.

10. In the FlexNet Beacon PowerShell Source Connection dialog, click **Save** to save the connection.
11. Select your new connection from the displayed list, and click **Schedule...**.
12. In the dialog that appears, select the name of your chosen schedule for inventory collection through this connection, and click **OK**.
13. At the bottom of the **FlexNet Beacon** interface, click **Save**, and if you are done, also click **Exit**.

Tip: *Consider whether you want to select your connection, and click* ***Execute Now****, before you exit.*

After a successful data import, the users, applications, licenses, and usage data are all visible in the appropriate pages of FlexNet Manager Suite.

Note: *To know more about the operations available on the* ***Inventory Systems*** *page of FlexNet Beacon, see Inventory Systems Page in the online help. For scheduling data imports through this connection, see Scheduling a Connection, also in help.*

XII

ServiceNow Integration with FlexNet Manager Suite

Using the FlexNet Manager Suite integration application for ServiceNow, you can exchange a limited set of data between FlexNet Manager Suite and ServiceNow to provide a consistent view of your hardware and software estate, and related contracts.

ServiceNow provides cloud-based IT Service Management, while FlexNet Manager Suite is focused on Software Asset Management. To help provide a unified view of your management data, there are two parts to the integration of these systems, each of which is available independently:

- Data on hardware assets, application installations, and contracts can be exported from FlexNet Manager Suite and imported into ServiceNow
- Data on assets and contracts can be exported from ServiceNow and imported into FlexNet Manager Suite.

This release of the ServiceNow integration application supports the following versions of ServiceNow:

- Kingston
- London
- Madrid
- New York.

Note: *ServiceNow has deprecated older versions of Transport Layer Security (TLS). Requires version 1.2.*

This ServiceNow documentation is divided into the following chapters:

- Key Concepts — key concepts to help you understand how and what type of data is merged with the integration application, where to view merged data, insight into how source of truth settings structure data into an optimized view with no duplicate or missing records
- Architecture, Components, and Prerequisites — background information including requirements you need to know to get started with the integration application, as well as contextual information to help you understand the configuration and operation of the integration
- Installation and Configuration — common tasks to be completed for all implementations, procedures for setting up

data flows from FlexNet Manager Suite to ServiceNow or from ServiceNow to FlexNet Manager Suite

- Operational Details — steps showing how the export of data from FlexNet Manager Suite works (to export hardware inventory, installed applications, and contract data), and steps showing how the export of data from ServiceNow works (to export asset and contract data)
- Appendices — integration property descriptions, a performance improvement tip to create additional indexes on your ServiceNow instance, procedure to remove a previously-installed legacy integration application, and more.

1

Key Concepts

Before using the **FlexNet Manager Suite for Enterprises integration application for ServiceNow**, it is helpful to understand some key concepts of the integration that are described in the following sections:

- Data Types that Can Be Merged — data types that can be exported from FlexNet Manager Suite and from ServiceNow
- How Data Is Merged — introduces the role that transform maps play in merging data between FlexNet Manager Suite and ServiceNow
- Source of Truth — how source of truth settings structure data into an optimized view with no duplicate or missing records
- Where to View Merged Data — where to go in FlexNet Manager Suite and ServiceNow in order to view data that has been merged by the integration application
- ServiceNow Computer and Application Records — table storage options in ServiceNow (Software Asset Management (SAM) Foundation plugin tables, Configuration Management Database (CMDB) tables, or both) and a description of how the integration application creates or updates computer records in ServiceNow and sets the CI classes.

Data Types that Can Be Merged

The following table shows the data types that can be exported from FlexNet Manager Suite and data types that can be exported from ServiceNow.

When exporting from:	The following data types can be exported:
FlexNet Manager Suite	Hardware inventory, installed applications, and contacts.
ServiceNow	Assets and contracts

Note: *When exporting from FlexNet Manager Suite, the three data types each correspond to a check box in the* ***ServiceNow export settings*** *section* ***Integrations*** *tab on the* ***System Settings*** *page. You can export any or all of these data types to ServiceNow. When exporting from ServiceNow, assets and contracts can be included in the export, as configured in scheduled jobs that can be accessed from the following ServiceNow menu:* ***FlexNet Manager Suite Scheduled Jobs****.*

How Data Is Merged

When data is merged between FlexNet Manager Suite and ServiceNow, transform maps are used to map the data. For more information, refer to Transform Maps for ServiceNow Integration.

When data is merged between ServiceNow and FlexNet Manager Suite, business adapters are used to map the data. For more information, refer to Business Adapter Mappings.

For both scenarios, an integration user needs to be created in ServiceNow so that both systems can communicate. For more information, refer to Creating a ServiceNow Integration User.

Source of Truth

When exporting from FlexNet Manager Suite for import into ServiceNow, a source of truth needs to be defined between systems in order to address any differences encountered during the merge. A set of integration properties are provided in ServiceNow (at **FlexNet Manager Suite > Integration Properties**) that let you specify whether to accept FlexNet Manager Suite as the source of truth for various data types. Choosing **No**> for any option toggles that setting to ServiceNow as the source of truth. For example, when merging inventory, if a record is not found with given criterion, a new record will be created if FlexNet Manager Suite is source of truth for **Adding Inventories**. For more information about source of truth integration properties, refer to Integration Properties.

Note: *For records of hardware inventory and installed applications, FlexNet Manager Suite is considered the authoritative source of truth.*

When exporting assets and contracts from ServiceNow for import into FlexNet Manager Suite, scheduled jobs need to be configured. For more information about scheduled jobs, refer to Setting Up Data Flows from ServiceNow to FlexNet Manager Suite. Then, the business adapters are used to import data into FlexNet Manager Suite. Assets and contracts are both set as the source of truth by default; however, you can edit the business adapters to change these settings. For more information, refer to Configuring FlexNet Beacon for Import.

Note: *For assets and contracts, ServiceNow is considered the authoritative source of truth.*

Where to View Merged Data

The following table shows where to go in FlexNet Manager Suite and ServiceNow to view data that is merged by the integration application. This not only helps you understand the integration but also helps you understand where to go in both systems in order to view and validate results.

Data type	FlexNet Manager Suite menu location	ServiceNow menu location
Hardware inventory	**Discovery & Inventory > All Inventory**	**FlexNet Manager Suite > Imported Records > Computers**

Data type	FlexNet Manager Suite menu location	ServiceNow menu location
Installed applications	**Applications** tab of inventory device properties.	**Software Installations** tab (SAM) or **Software Installed** tab (CMDB) of a computer record
Contracts	**Procurement > All Contracts**	**FlexNet Manager Suite > Imported Records > Contracts**

You may also want to compare inventory device properties.

Property	FlexNet Manager Suite	ServiceNow
General properties of inventory device such as: device type (Computer, Virtual machine), Manufacturer, Serial number, etc.	**Inventory device properties** (accessed by selecting a record from **All Inventory**)	**Computer** record (accessed by selecting a record from **FlexNet Manager Suite > Imported Records > Computers**)

ServiceNow Computer and Application Records

Installed Applications

When exporting installed application data from FlexNet Manager Suite for import into ServiceNow, you have the option to select which tables store the data. Prior to integration application v4.0, installed application records were stored into Software Asset Management (SAM) Foundation plugin tables. Integration application v4.0 and later gives you the option to store imported application records into Configuration Management Database (CMDB) tables, SAM tables, or both. A new integration property, **Use CMDB and/or SAM tables for installation**, has been added to the **Integration Properties** page. As a result of this enhancement, the SAM Foundation plugin is no longer a requirement for using the integration application. For more information, refer to Integration Properties.

Hardware Inventory

When hardware inventory records are exported from FlexNet Manager Suite to ServiceNow, the integration creates or updates computer records in ServiceNow. The integration also sets the correct CI class on these records based on the **Inventory Device Type** and **Operating System**. This enables viewing these records in the respective views within ServiceNow. The following table shows the logic of how the CI class of computer records are classified in ServiceNow and also shows where to view the resulting records in ServiceNow.

Table 2: Hardware inventory exported from FlexNet Manager Suite and imported to ServiceNow

Inventory device type of Inventory record in FlexNet Manager Suite	Operating system of Inventory record in FlexNet Manager Suite	CI class of Computer record in ServiceNow	View in ServiceNow
VM Host	contains "Microsoft" or "Windows"	Hyper-V server	**Configuration > Hyper-V > Hyper-V Servers**
	contains "VMware"	ESX server	**Configuration > VMware > ESX Servers**
	anything else	Visualization server	no direct link, but you can enter the following into the ServiceNow Filter navigator to access view: `cmdb_ci_virtualization_server.list`
Computer Virtual Machine Mobile Device	contains "AIX"	AIX Server	**Configuration > Servers > AIX**
	contains "Solaris" or "Sun OS"	Solaris Server	**Configuration > Servers > Solaris**
	contains "OS X"	OS X	**Configuration > Servers > OS X**
	contains "HP-UX"	HPUX Server	**Configuration > Servers > HPUX**
	contains "Unix"	Unix Server	**Configuration > Servers > Unix**
	contains "Linux"	Linux Server	**Configuration > Servers > Linux**
	contains "Microsoft" or "Windows" and contains "Server"	Windows Server	**Configuration > Servers > Windows Servers**
	anything else	Computer	**Configuration > Base Items > Computer**

For all the virtual machine records in an export, the import creates or updates Virtual Machine Instance records in ServiceNow and sets the correct CI class on these records based on the virtual machine type. The following table shows the logic of how the CI class of virtual machine records are classified in ServiceNow and also shows where to view the resulting records in ServiceNow.

Table 3: Virtual machines exported from FlexNet Manager Suite and imported to ServiceNow

VM Type of Inventory record in FlexNet Manager Suite	CI class of Virtual Machine Instance record in ServiceNow	View in ServiceNow
AWS EC2	EC2 Virtual Machine Instance	no direct link, but you can enter the following into the ServiceNow Filter navigator to access view: `cmdb_ci_ec2_instance.list`
Hyper-V	Hyper-V Virtual Machine Instance	**Configuration > Hyper-V > Virtual Machine Instances**

VM Type of Inventory record in FlexNet Manager Suite	CI class of Virtual Machine Instance record in ServiceNow	View in ServiceNow
VMware	VMware Virtual Machine Instance	**Configuration > VMware > Virtual Machine Instances**
Zone	Solaris Virtual Machine Instance	no direct link, but you can enter the following into the ServiceNow Filter navigator to access view: `cmdb_ci_solaris_instance.list`
LPAR	Virtual Machine Instance	**FlexNet Manager Suite > Imported Records > Virtual Machines**
nPar		
Oracle VM		
SRP		
vPar		
WPAR		
Unknown		

Tip: *Integration properties allow you to choose whether to use FlexNet Manager Suite or ServiceNow as the source of truth to create and update records. Another property allows you to choose whether FlexNet Manager Suite or ServiceNow should be used as the source of truth to update CI classes. See Adding Inventories, Updating inventories, and Updating inventory class name (sys_class_name) in Integration Properties.*

2

Architecture, Components, and Prerequisites

This chapter provides information you need to get started with the integration application.

The information is divided into the following sections:

- Architecture — architectural diagrams to help you understand both configuration and operation of the integration between ServiceNow and FlexNet Manager Suite
- Prerequisites — requirements for integrating FlexNet Manager Suite and ServiceNow
- Download Adapter Tools Archive — how to download the Adapter Tools archive that includes content needed for many adapters.

Architecture

This big-picture overview provides context for understanding both configuration and operation of the integration between ServiceNow and FlexNet Manager Suite.

Because ServiceNow is cloud-based, and your implementation of FlexNet Manager Suite is on your premises, there is a level of separation required (for example, the products must be accessed in separate browser tabs/windows). The following architecture allows for the necessary data flows between these two distinct systems.

The following two processes (illustrated in the diagram) operate completely independently:

- The export of computer, application, and contract data from FlexNet Manager Suite into ServiceNow (purple arrow). For details, see Process for Exports from FlexNet Manager Suite to ServiceNow.
- The export of computer and contract data from ServiceNow for import into FlexNet Manager Suite (gold arrows). For details, see Process for Exports from ServiceNow to FlexNet Manager Suite.

Prerequisites

The following are requirements for integrating FlexNet Manager Suite release 2019 R2 and ServiceNow. At a high level, you need:

- A functional implementation of FlexNet Manager Suite
- An operational ServiceNow instance
- Installation of the FlexNet Manager Suite integration application.

The following provides further details for each of the above three required components:

- You need a functional implementation of FlexNet Manager Suite.

  Important: *To take full advantage of the new features included in the v4.0 integration application, it is recommended that you run FlexNet Manager Suite 2019 R1 or later. For more information about limitations with earlier releases of FlexNet Manager Suite, see the knowledge article Feature Support for ServiceNow Integration, in the Flexera Customer Community.*

  - You must have a license issued by Flexera that permits use of the integration application. This license authorizes all communications in both directions through the integration application. To check, in the web interface navigate to the system menu (⚙ ▾ in the top right corner) and click **FlexNet Manager Suite License**. Check under the **License details** for `ServiceNow integration enabled: Yes`. If this is not the case, request the license from your Flexera representative.

- You must have an operational inventory beacon that communicates with your central application server for FlexNet Manager Suite.
- The FlexNet Manager Suite feature called ServiceNow Exporter must be installed. To check:
  1. On your batch server, navigate to the installation directory for FlexNet Manager Suite.
  2. Navigate down to the child directory `DotNet\bin`.
  3. Validate that there is a `ServiceNowExport` folder, with content. If so, the check is satisfactorily completed.
  4. If the folder is missing, you must re-reun the FlexNet Manager Suite installer, and ensure that the `ServiceNow Exporter` feature is selected for installation. You should also re-reun the configuration PowerShell script in order the create the required scheduled task. (For more information, see your copy of the *Installation Guide* PDF, available through the title page of online help.)

- You must have an operational ServiceNow instance.
  - For data flows from ServiceNow to FlexNet Manager Suite, you need a MID server configured for your ServiceNow implementation.

    Tip: *If you prefer, your FlexNet Beacon and ServiceNow MID server can be implemented on the same physical computer (provided that there are common communications requirements — for example, you don't have a case where one requires a proxy server setting that the other cannot use).*

  - For data flows from FlexNet Manager Suite to ServiceNow: If your ServiceNow instance has the Software Asset Management (SAM) Foundation plugin installed, you can choose to direct installed application data to SAM tables, CMDB tables, or both. If your ServiceNow instance does not have the SAM Foundation plugin installed, you must choose to direct Installed Application data to CMDB tables. For details on both, see Configuring the Software Asset Management Foundation Plugin and see the description of the **Use CMDB and/or SAM tables for installation** property in Integration Properties.
  - This release of the ServiceNow integration application supports the following versions of ServiceNow:
    - Kingston
    - London
    - Madrid
    - New York.

    Note: *ServiceNow has deprecated older versions of Transport Layer Security (TLS). Requires version 1.2.*

- If this is your first ever install of the FlexNet Manager Suite integration application for ServiceNow, you must install it from the ServiceNow Store (see Installing the FlexNet Manager Suite Integration Application from the ServiceNow Store), and then any updates to the app are also done through the ServiceNow store. Alternatively, if you have previously installed an earlier version of the integration using an update set, refer to the knowledge article Upgrading ServiceNow Integration Application Using an Update Set in the Flexera Customer Community.
- For first time users, and if you will be exporting data from ServiceNow to FlexNet Manager Suite, two business adapters (`ServiceNowAssets.xml` and `ServiceNowContracts.xml`) must be used for assets and contracts. These can be found in the in the unzipped adapter tools archive (see Download Adapter Tools Archive). For more

information, refer to Configuring FlexNet Beacon for Import.

Download Adapter Tools Archive

The Adapter Tools archive includes content for many adapters, and is updated on the Flexera website from time to time.

Start this procedure using a web browser on a computer that has good network accessibility from all the machines needing installations for your adapter.

To download the adapter tools archive:

1. Download the `Adapter Tools for` FlexNet Manager Suite `2019 R2.zip` archive from the Flexera Customer Community knowledge base:

   a. Access https://flexeracommunity.force.com/customer/articles/en_US/INFO/Adapter-Tools-for-FlexNet-Manager-Suite.

   Tip: *Access requires your Customer Community user name and password. If you do not have one, use the link on the login page to request one.*

   b. Click the link `Adapter Tools for FlexNet Manager Suite`.

   A new browser tab may appear temporarily, and the download of `Adapter Tools for` FlexNet Manager Suite `2019 R2.zip` commences.

   c. In your browser dialog, choose to save the file, and if the browser allows it, direct the saved file to a convenient working location (such as `C:\temp` on a central, accessible server).

   If your browser saves the file to a default location (such as your `Downloads` folder), move or copy it to the appropriate working location when the download is finished.

2. Right-click the zip archive, and choose **Extract All...**.

The folders are now available for the range of adapters in the Adapter Tools archive.

3

Installation and Configuration

Because the system involves a number of different servers, there are several installation and configuration steps. These are described in full in the following topics.

The information is divided into the following sections:

- Setting Up the Integration — common tasks that must be completed for all implementations
- Setting Up Data Flows from FlexNet Manager Suite to ServiceNow — required if you want to export the FlexNet Manager Suite data on hardware assets, installed applications and contracts for use within ServiceNow
- Setting Up Data Flows from ServiceNow to FlexNet Manager Suite — required if you want to import assets and/or contracts data into FlexNet Manager Suite.

Setting Up the Integration

Most of the groundwork on the FlexNet Manager Suite side has already been completed as part of product installation. Therefore the balance of the work that is common to both directions of data flow involves setting up ServiceNow.

- In ServiceNow terminology, you need to install an integration 'application' in ServiceNow (see Installing the FlexNet Manager Suite Integration Application from the ServiceNow Store).
- There must also be a user account correctly configured in ServiceNow (see Creating a ServiceNow Integration User).

Installing the FlexNet Manager Suite Integration Application from the ServiceNow Store

The steps in this section provide steps for a **first time install** of the **FlexNet Manager Suite for Enterprises** integration application. If you have previously installed an earlier version of the integration using an update set, instead refer to the knowledge article Upgrading ServiceNow Integration Application Using an Update Set in the Flexera Customer Community (login required).

To install the FlexNet Manager Suite for Enterprises integration application from the ServiceNow store:

1. Log into ServiceNow Store with ServiceNow HI credentials.
2. In the **Search** box, enter **FlexNet Manager Suite for Enterprises** and press **Enter**.

   Application information appears in the search results.
3. From the search results, click **FlexNet Manager Suite for Enterprises**.

   Information about the application displays.
4. In the right pane of the application window, view the Compatibility section to ensure that the integration application is compatible with the version of ServiceNow that you require. If the version of ServiceNow that you require is not listed, contact FlexNet Manager Suite Technical Support.
5. In the right pane of the application window, click the **Get** button.

   A notice appears.
6. At the bottom of the notice, click **Continue**.
7. On the **Purchase of FlexNet Manager Suite for Enterprises** page, choose the **Make available on specific instances** radio button. and click the **Select** button.

   A **Select Instances** dialog appears.
8. In the **Available Instances** pane, double-click the instance that you want the app to be available on and click **OK**.
9. Select the **I accept** check box (to accept the ServiceNow Terms of Use and Vendor App Subscription Terms and Conditions) and click **Get**.
10. Log into the ServiceNow instance where you made the integration application available. (You selected this instance in **Step 7**)
11. In the Filter navigator, enter **System Applications** and then under **System Applications**, click **Applications**.

    The **Applications** list appears.
12. Click the **Downloads** tab.

    The **Downloads** list appears.
13. Scroll to the **FlexNet Manager Suite for Enterprises** application, and click the **Install** button.

The application now appears in the ServiceNow menu, displayed in the left-hand navigation panel as **FlexNet Manager Suite**.

Creating a ServiceNow Integration User

This simple process sets up the user account used for integration between Service Now and FlexNet Manager Suite, and assigns the user to the appropriate role. Complete this task while you are still logged into ServiceNow as an administrator.

To create the user account in ServiceNow:

1. In ServiceNow, go to **User Administration > Users**.
2. Click **New**, complete the properties for your new user, and click **Submit**.

   Tip: *Take note of the user ID and password, which you will need again later in the set-up processes.*

3. Once the creation process finishes, click the hyperlinked **User ID** for this user, and scroll down to **Roles**.
4. Click **Edit**, and select `x_fls_flexera_fnms.admin` from the collection, and add to **Roles List**.
5. Click **Save**.
6. Close the user properties page.

Setting Up Data Flows from FlexNet Manager Suite to ServiceNow

For records of installed applications and hardware inventory, FlexNet Manager Suite is considered the authoritative source of truth. Data on installed applications and hardware inventory, as well as contracts, can be exported from FlexNet Manager Suite to ServiceNow. The summary of the export process is:

1. On the batch server for FlexNet Manager Suite, a Microsoft scheduled task `Export to ServiceNow` initiates the process.
2. The scheduled task launches an exporter utility for FlexNet Manager Suite which extracts the appropriate data and transfers it to ServiceNow using API calls. This communication relies on *all* of:
   - The URL of your ServiceNow instance
   - The user account registered for accessing ServiceNow
   - The communications token copied from ServiceNow and saved in FlexNet Manager Suite.
3. In ServiceNow, the FlexNet Manager Suite integration application processes the data received through the API calls, and schedules imports of that data using the ServiceNow Import Set.

   Tip: *The Transform Maps used during import to switch data from one product's formats to the other are also defined in the integration application.*

If you wish to make use of this data flow, you need to complete the following processes:

- Configuring ServiceNow for Import
- Configuring FlexNet Manager Suite for Export
- Optionally, you may customize the export from FlexNet Manager Suite using the guidelines in Configuring the Utility for Export from FlexNet Manager Suite.

Configuring ServiceNow for Import

Continue the following while still (or again) logged into ServiceNow with administrator privileges.

To configure imports into ServiceNow:

1. In the **FlexNet Manager Suite** group in the navigation bar, click **Generate Token**.
2. From the account details you created earlier (see Creating a ServiceNow Integration User), insert the user ID in the **Username** field and supply the **Password**, and click **Generate OAuth Token**.

   A long `Refresh Token` appears at the top of the page.
3. Copy this `Refresh Token` value, and transfer to the web interface of FlexNet Manager Suite.
4. When doing Installed Application data export, decide what tables in ServiceNow that you want this data to go in. To do this: In ServiceNow, navigate to **FlexNet Manager Suite > Integration Properties** and then in that page, from the **Decide whether too use CMDB or SAM tables** drop-down list and select one of the following options:
   - **Decide automatically** — Import to SAM tables. If no SAM tables exist, data is instead imported to CMDB tables.
   - **Use CMDB tables** — Import to CMDB tables.
   - **Use SAM tables** — Import to SAM tables. If there are no SAM tables, the import will fail.
   - **Use CMDB and SAM tables** — Import to CMBD tables **and** SAM tables. If SAM tables do not exist, the import will fail.
5. Change the account used for processing the import in ServiceNow. To do this:
   a. Go to **FlexNet Manager Suite > Advanced > Scheduled Data Imports**.

      The Scheduled Data Imports list appears.

   b. Click **FNMS Import Schedule**.

   c. Change the **Run as** user to the integration user. (This is the user ID that you created earlier (see Creating a ServiceNow Integration User).

   d. Click **Update**.

Configuring FlexNet Manager Suite for Export

You need to complete this configuration once to commence operations. Repeat if your ServiceNow details change at any time in the future. You may also use this same page to trigger an immediate export from FlexNet Manager Suite to ServiceNow.

To configure export from FlexNet Manager Suite:

1. Ensure that you are (still) logged into FlexNet Manager Suite with an account that is a member of the `Administrator` role, and that this account has the **Configure** access right.

2. In the web interface, navigate to the system menu (⚙ ▼ in the top right corner) > **System Settings**, select the **Integrations** tab, and expand the **ServiceNow** accordion.

   Tip: *The **Integrations** tab is only visible when your account has the access rights described above.*

3. In **Instance URL**, enter the protocol and path to your ServiceNow website (for example, `https://YOUR-INSTANCE.service-now.com`).

4. In **Username**, enter the account you use to access ServiceNow. This must exactly match the **User ID** value shown in the ServiceNow interface (see Creating a ServiceNow Integration User).

5. Paste the **Token** value from ServiceNow into the **Token** field on this page (see Configuring ServiceNow for Import).

   Tip: *The **Username** is mandatory for FlexNet Manager Suite to commence the export process, but it is not transferred back to ServiceNow, as the **Token** value alone is now sufficient for authentication.*

   Note: *In future, if you wish to temporarily disable these data exports from FlexNet Manager Suite, you can either disable the Windows scheduled task `Export to ServiceNow` or remove any one or more values from **Instance URL**, **Username**, or **Token**.*

6. If necessary, change the default selection of data types to export to ServiceNow. It is best practice to export all available data types.

   Warning: *Hardware inventory details are critical to the ServiceNow data set. Contracts and installed applications both have dependencies on assets in that system. If you clear the **Hardware inventory** check box, you may produce gaps in the contract and software records imported into ServiceNow.*

7. Click **Save**.

   When details are complete, the export from FlexNet Manager Suite to ServiceNow is triggered weekly at 3am on Sunday mornings by default (if required, you may edit the Windows scheduled task `Export to ServiceNow` to modify the schedule). If you need to trigger an additional export (for example, when commencing integrated operations), continue with the rest of this process.

   Tip: *While the first export of any given data type is always a full export of all active records, subsequent data transfers from FlexNet Manager Suite to ServiceNow are differential (that is, only data that is new or changed since the last successful export of the same type is included). If you wish to transfer all available active data (rather than the differential data set), select the **Perform full export** check box before continuing.*

8. Click **Export** to trigger an immediate export using the settings saved in the fields displayed above.

   The **Export** button, and the scheduled task, trigger the command-line executable `fnmp_servicenow_export.exe`, which:

   - Extracts the required data of the first data type from your operations database (the normal order when all are selected is hardware inventory, contracts, installed applications)
   - Segments the data for easier transmission across the Internet, minimizing time-out risks with ServiceNow
   - Reassembles the data into staging tables in ServiceNow

- Repeats the process for the next selected data type.

Separate transforms on your ServiceNow instance then map each data type from the staging tables to your operational ServiceNow tables.

Tip: *It is also possible to run the executable from the command line, provided you have a copy of the ServiceNow token available (see Command-Line Tool for Export to ServiceNow).*

To verify progress of a test run:

1. In the ServiceNow navigation bar, access **FlexNet Manager Suite > Integration History > Import Transactions**. Each of these transactions represents a data chunk* or segment transferred to ServiceNow. The segmentation is necessary because of the large number of records that may need to be transferred. When the data is received by ServiceNow, the integration application separately schedules each transaction for import into the operational database tables in ServiceNow. As each transaction is completely imported, its **State** column value is set to `Succeeded`.
2. Similarly, access **FlexNet Manager Suite > Integration History > Import Runs**. Each data type exported from FlexNet Manager Suite creates a separate entry here for import into ServiceNow. For each data type, when all its individual transactions have succeeded, the **State** value on the overall Import Run is also set to `Succeeded`.
3. Finally, when all the transferred data types have been successfully imported, a status report is sent to FlexNet Manager Suite. On the **System Settings** page in the **Integrations** tab, the status is displayed under **Last completed export**.

* The default configuration for data transfers from FlexNet Manager Suite to ServiceNow uses these values:

- The number of database records included in each transferred segment: 4000
- The number of retries if ServiceNow returns a connection failure: 10
- The length of time to wait for ServiceNow to respond before timing out: 30 seconds
- The maximum number of records of each data type to be included in the transfer, by type:
  - Hardware inventory: 10,000,000
  - Contracts: 500,000
  - Installed applications: 10,000,000.

For details of configuring these additional elements in an on-premises implementation, see Configuring the Utility for Export from FlexNet Manager Suite.

Configuring the Utility for Export from FlexNet Manager Suite

Optionally, the configuration file for the export utility may be customized to suit your environment.

Several settings for the utility for exporting data from FlexNet Manager Suite are configurable, including:

- The maximum number of records for each data type (hardware inventory, installed applications, and contracts) that may be transferred to ServiceNow in each export

- The maximum number of records in each data segment or 'chunk' transferred
- Log file location
- Maximum retries after failure of a web request.

All reside in one configuration file, called `fnmp_servicenow_export.exe.config`, located in the same folder as the executable `fnmp_servicenow_export.exe` (which performs the data export from FlexNet Manager Suite to ServiceNow) on your batch server (or, in a smaller implementation, the server hosting the equivalent functionality). The default path is *`installation-directory`*`\DotNet\bin\ServiceNowExport`.

To configure exports from FlexNet Manager Suite:

1. Save a backup copy of the configuration file, for example as `fnmp_servicenow_export.exe.config.orig`, so that you can revert if there are problems.
2. Open the configuration file `fnmp_servicenow_export.exe.config` in your preferred XML or plain text editor. (Do not edit in a word processor that cannot save plain text files.)
3. Optionally, customize the values for data segmentation, retries, and timeout when the executable is accessing ServiceNow.

Path/element	Attribute	Default	Notes

`<configuration>`
`  <fnmpGroup>`
`  <fnmp>`
`  <baseConfig />`
	`maxRetries`	`"10"`	When ServiceNow is not responding, the executable will retry each connection (one for each data segment being transferred) this many times. If this number of retries is reached for any data segment, the export fails. Each successful connection resets the count to zero. (If this attribute is omitted, the executable uses a default value of 10.)

`<configuration>`
`  <fnmpGroup>`
`  <fnmp>`
`  <baseConfig />`
	`timeout`	`"30000"`	The maximum number of milliseconds to wait for a response from ServiceNow. This timer is applied independently to each connection request (at least one per data segment being transferred). If ServiceNow does not respond within this interval (default 30 seconds), a new request is issued (up to `maxRetries`). (If this attribute is omitted, the executable uses a default value of 30000.)

Path/element	Attribute	Default	Notes

`<configuration>`
`  <fnmpGroup>`
`  <fnmp>`
`  <application />`
	`maxRows`	`"10000000"`	Specifies the maximum number of application records that can be exported in a single export.

`<configuration>`
`  <fnmpGroup>`
`  <fnmp>`
`  <contract />`
	`maxRows`	`"500000"`	Specifies the maximum number of contract records that can be exported in a single export.

`<configuration>`
`  <fnmpGroup>`
`  <fnmp>`
`  <inventory />`
	`maxRows`	`"1000000"`	Specifies the maximum number of hardware inventory records that can be exported in a single export.

`<configuration>`
`  <fnmpGroup>`
`  <fnmp>`
`  <records />`
	`numRecords`	`"4000"`	The number of records in each XML-based data segment to be transferred from FlexNet Manager Suite to ServiceNow. To avoid time-out issues with ServiceNow, the total exported data (in each of the three available types) is chunked into segments containing a maximum of this number of records.

4. Optionally, configure the location for your log files.

Path/element	Attribute	Default	Notes

`<configuration>`
`  <log4net>`
`  <appender>`
`  <file />`
	`value`	`"C:\Temp\Log\ServiceNow_`*`date`*`.log"`
		(All on one line)
			You may adjust the path to store the log files to suit your preference. For on-premises installations, you may also change the file name.

With the other default values for the elements under appender, multiple passes on the same date append to the end of the log file, and a new log file is commenced each day.

Setting Up Data Flows from ServiceNow to

FlexNet Manager Suite

Details of hardware assets and/or contracts, for which ServiceNow is considered the authoritative source of truth, can be imported into FlexNet Manager Suite from ServiceNow. In summary, the process is:

1. Each data collection is triggered by a scheduled job in ServiceNow that invokes a script.
2. The script uses ServiceNow database views to gather the required data and transfers it, in chunks, to the appropriate MID server.
3. The MID server writes the chunks into an XML file.
4. A separate schedule configured on the inventory beacon for FlexNet Manager Suite triggers an import through a business adapter, collecting the XML file, transforming it, and uploading it to the central application server.
5. The normal import processes on the application server imports the data (transformed as appropriate) into the operations databases for FlexNet Manager Suite.

If you wish to make use of this data flow, you need to complete the following processes:

- Setting Up a MID Server
- Configuring ServiceNow for Export
- Configuring FlexNet Beacon for Import.

No configuration of the central application server is needed, since this part of the integration leverages all standard operating processes.

Setting Up a MID Server

You may already have an operational MID server (accessible from your chosen inventory beacon) that you wish to use for data transfers from ServiceNow to FlexNet Manager Suite. If so, and the MID server is validated in ServiceNow, you may skip this topic.

On the other hand, you may prefer to set up a specific MID server especially for this integration. If so, follow these steps while you are still logged into ServiceNow with administrator privileges.

To set up a MID server that stages data from ServiceNow:

1. In the navigation bar, expand **MID Server** and click **Downloads**.
2. Click the link appropriate for your platform.
3. In the navigation bar, in the **MID Server** group, click **Installation Instructions**, and click the link appropriate to your version of ServiceNow.

   The documentation opens in a new tab.
4. Step through those instructions to complete installation of your MID server.

   Take particular note of the exact name you give this MID server, as you will need it again soon. (It is also available through **MID Server > Servers**.)
5. Be sure to validate your MID server (see the **Validate the MID Server** link on the MID Server installation page).

Configuring ServiceNow for Export

Now that you have a MID server installed, two further configuration points for ServiceNow remain:

- Identifying the MID server as the one to use for the exports to FlexNet Manager Suite
- Scheduling the exports from ServiceNow.

Continue the following while still logged into ServiceNow with administrator privileges.

To configure ServiceNow for export:

1. In ServiceNow, go to **FlexNet Manager Suite > Integration Properties**.
2. In the list of **Integration Properties**, scroll down to the **MID server name** field, and enter the full name of the MID server.
3. Optionally, in the **Export file path** field, enter an alternative file path location to store the XML file (instead of the default MID server location). Because the FlexNet Beacon reads this file using the business adapter, FlexNet Beacon needs access to this file location.
4. Scroll down (if necessary), and click **Save**.
5. In ServiceNow, go to **FlexNet Manager Suite > Scheduled Jobs**.

   The list of standard jobs to export data from ServiceNow is displayed, including:

   - `Export Assets from ServiceNow`
   - `Export Contracts from ServiceNow`.

   Each of these jobs needs to be configured for your environment.
6. Add a **Run as** column to this page and enter the integration user ID in the column for both jobs. To do so:
   a. Click the Personalized List (gear) icon in the upper left corner at the top of list of scheduled jobs.
   b. On the **Personalize List Columns** dialog, select **Run as** from the **Available** columns and move it to the **Selected** columns.
   c. On the Scheduled Script Executions page, change the **Run as** user (that is blank by default) to the integration user ID (see Creating a ServiceNow Integration User).
7. Click the first export task to schedule (such as `Export Assets from ServiceNow`).

   The **Scheduled Script Execution** properties are displayed.

   Note: *The MID server collects the exported XML data in chunks, such that the XML file is not well formed until the last chunk has been received. Be sure to schedule sufficient time for the exports to complete before the scheduled import from the MID server into FlexNet Manager Suite.*
8. Change the value in the **Run** field to your desired schedule.

   These data exports may be large, based on the size of your ServiceNow repository. A common practice would be to choose `Weekly` exports.

> ***Tip:*** *ServiceNow by default limits its exports from the database to 10,000 records (for a general discussion, see https://community.servicenow.com/community/blogs/blog/2014/08/01/increasing-the-export-limit, but note that this page discusses limits for exports from the ServiceNow user interface grids, which are not relevant to the database exports to FlexNet Manager Suite). You can modify the maximum number of records exported from ServiceNow by having your ServiceNow administrator add the property* `glide.db.max_view_records` *and setting a new limit. For instructions on adding the property, see http://wiki.servicenow.com/index.php?title=Adding_a_Property#gsc.tab=0.*

9. Use the additional fields that appear to complete configuring your scheduled job.

   For example, for `Weekly` exports, set the **Day**, and **Time** of day, when the export should occur. An off-peak time is preferable.

10. Click **Update** to save your changes.

    Repeat the scheduling for the `Export Contracts from ServiceNow` task.

11. With both exports now scheduled, run a test of each by clicking **Execute Now** on the associated **Scheduled Script Execution** properties page.

    (This same button can be used to run the export if you chose the **Run** `On Demand` setting for the schedule.) Use the test to assess the time taken to complete the export and finish a well-formed XML file on the MID server, as a check on your scheduling plans. The test also allows you to troubleshoot any problems.

12. Monitor overall progress in ServiceNow by going to **FlexNet Manager Suite > Integration History > Export Runs**.

    The **State** column on this view shows the progress of export, and the **Stage** column displays `File Completion` when the XML file is done.

13. On the MID server, check for well-formed (complete) XML files.

    The file names are of the form `x_fls_flexera_fnms_`*`ExportType`*`.xml`, where the placeholder *`ExportType`* is either `Asset` or `Contract`.

This completes the configuration of ServiceNow for data flows from ServiceNow to FlexNet Manager Suite. These same data flows also require some configuration of FlexNet Manager Suite to receive the data. This side of the configuration is covered next.

Configuring FlexNet Beacon for Import

If you are not already familiar with importing business-related data to FlexNet Manager Suite, the following terminology summary may help:

- A "business adapter" is an XML file that allows configuration of a connection to a source of business data, as well as mapping of that data from the source format to the database tables and columns required in FlexNet Manager Suite. The data from ServiceNow is handled as business data, and therefore data transfers make use of business adapters.
- The "Business Adapter Studio" is a utility available on each inventory beacon for customizing and testing business adapters.
- The "Business Importer" is the process that exercises the business adapters to the schedule you define, and

controls each import as defined in its business adapter.

In previous tasks, you have configured ServiceNow to run two exports that queue data about hardware assets and contracts and send it to the MID server of your choice. On the MID server, the data is written into XML files. Once these have been completed, two business adapters running on your inventory beacon collect this data and upload it to your central operations databases. You must use the following process to configure these business adapters before they can operate.

Tip: *Clearly, the inventory beacon and the ServiceNow MID server must access a common folder for the transfer of the XML files. It is possible that the FlexNet Beacon software and the ServiceNow MID server are installed on the same physical machine; but if they are on separate machines, network access allowing a file share is required. This means you may need to grant at least read access for the share on your MID server to two accounts on your inventory beacon:*

- *The local administrator account with which you will log in now for configuring and testing your business adapters*
- *The account that runs your FlexNet Beacon Service, which will exercise the business adapters in production (by default, this is the local SYSTEM account on your inventory beacon).*

As well as access to the MID server, your inventory beacon requires access to the unzipped archive you downloaded for the ServiceNow integration application to complete this process.

To configure an inventory beacon for importing data for FlexNet Manager Suite:

1. On your chosen inventory beacon, log in as a local administrator.
2. If necessary, validate that, from this server, you can access the file share where the MID server saves the completed XML files.
3. In the unzipped archive you downloaded for the ServiceNow integration application, open the `Importer` folder and locate the two business adapters `ServiceNowAssets.xml` and `ServiceNowContracts.xml`.
4. Copy these two files to the business adapters folder, *`%CommonAppData%`*`\Flexera Software\Beacon\BusinessAdapter`, on the inventory beacon.
5. Start the FlexNet Beacon software from the Windows start menu on the inventory beacon, using the **Run as administrator** option.
6. In the interface for FlexNet Beacon, select the **Business Importer** page from the **Connections** group (in the left-hand navigation pane).

   The two adapters appear in the list of available business adapters.
7. For each of these adapters in turn, select the adapter from the list, and click **Edit...**. In the following dialog, click **Edit Adapter**.

   The Business Adapter Studio opens, editing this adapter.
8. In the Business Adapter Studio, select the top level import node (like `ServiceNowAssets` or `ServiceNowContracts`) and locate the **File name** field. Complete the file path and file name for the appropriate XML file on the MID server. If you have already exported data from ServiceNow to the MID server, you may click the **...** button and browse to the respective XML file in the folder on the MID server.

   The following example from the assets adapter shows a path when the MID server and inventory beacon are

installed on the same physical computer:

```
C:\ServiceNow\MyMIDServerFolder\agent\x_fls_flexera_fnms_asset.xml
```

The following example from the contracts adapter shows a path when the MID server is separate, using UNC format:

```
\\MyMIDServer\ServiceNow\MyMIDServerFolder\agent\
x_fls_flexera_fnms_contracts.xml
```

9. When the file path and name are identified, click **Refresh** (just below the **...** button).
10. **Save** the modified adapter; and repeat the process for the other adapter.
11. When both are completed and saved, close Business Adapter Studio.
12. If you have not already created the schedule(s) on which you want to run these adapters, switch to the **Scheduling** page of the FlexNet Beacon interface, and create one or two now.

    You may run these two business adapters on the same schedule, as the inventory beacon queues them and runs them one after the other.

    Important: *Remember that there is a delay between when the scheduled job starts executing in Service Now, and when the MID server finishes assembling the transmitted chunks into a valid XML file. You can test the elapsed time for your environment by triggering each export, and waiting until, in the* ***Export Runs*** *view (in ServiceNow), it reaches the* `Succeeded` *state. Allow a safety buffer for future data growth, and set the schedule for the business adapters to run after the exports and assembly of the XML files are completed.*

13. In the **Business Importer** page of the FlexNet Beacon, select each adapter in turn, and:
    a. Click **Schedule...**.
    b. From the list of available schedules, select the one for each adapter.
    c. Click **OK**.
    d. Ensure that the adapter is marked as **Enabled**.
    e. Repeat for the other adapter.
14. At the bottom of the inventory beacon interface, click **Save** to store your changes.
15. Optionally, if you have test data waiting for a system check, you can select each adapter in turn, and click **Execute Now**.

    The Business Importer takes the raw data from XML files, transforms it into the formats required for FlexNet Manager Suite, and uploads it to the central application server.

To check the status of imports, in the web interface of FlexNet Manager Suite, navigate to the system menu (⚙ ▼ in the top right corner) and choose either **Data Inputs** (in the **Business Data** tab) or **System Health > System Tasks**.

Tip: *If you need to interrupt the export from ServiceNow, disable the scheduled jobs in ServiceNow, and disable the relevant schedule(s) in FlexNet Beacon.*

4

Operational Details

The exchange of data between ServiceNow and FlexNet Manager Suite happens in a pair of independent processes driven by schedules (either Windows scheduled tasks or schedules built in to the products).

As these schedules by default run on a weekly cycle, you may expect the data to be synchronized after the weekly runs and up until either system is subsequently updated. If you require an intermediate synching after an important update to one system or the other, you can also trigger either process manually.

The process flows are described in detail in the following topics. Notes about configuration and customization are included.

For your on-premises implementation of FlexNet Manager Suite, the command-line utility is also documented here, should you choose to run that directly.

Process for Exports from FlexNet Manager Suite to ServiceNow

A Windows scheduled task `Export to ServiceNow` triggers exports to ServiceNow at 3am every Sunday morning.

If you are in an administrator role, you can also trigger an export through the web interface for FlexNet Manager Suite. Navigate to the system menu (⚙ ▼ in the top right corner), choose **System Settings > Integrations** and expand the **ServiceNow** section.

Process Details

1. On the central application server, when the export is triggered, a utility is launched that:
   - Checks whether you have licensed the ServiceNow integration option from Flexera (if not, it shows an error in the same page of the web interface).
   - Tests your ServiceNow credentials in the **ServiceNow export settings** section of **System Settings > Integrations**.
2. Next, the utility checks whether ServiceNow is able to accept an import at this time:
   - It checks whether a record in `Scheduled Data Imports`, and another in `Data Sources`, have been created

by the integration application. (Links to the relevant pages where you can inspect the records in ServiceNow are under **FlexNet Manager Suite > Advanced**.)

- It ensures that no prior import is already in process, checking that there is *no* record in the `Import Runs` table or the `Import Transactions` table that are incomplete (that is, have a `State` other than `Succeeded` or `Failed`). Both lists are available for inspection in the **FlexNet Manager Suite Integration History** group.

If either of the checks fails, the utility abandons the export and displays an appropriate error.

3. The utility checks the connection with the operations databases for FlexNet Manager Suite. (If there is any failure, the utility abandons the export and displays an appropriate error.)
4. All being well, the data export, transfer, and import processes are run for all data types (unless any of them have been specifically excluded) in the following order:
   a. Hardware inventory
   b. Contracts
   c. Installed applications

   Caution: *In ServiceNow, contracts and installed applications both have dependencies on hardware inventory. It is recommended that you do not exclude the hardware inventory, as this may result in unpredictable gaps in other records when the correct dependencies cannot be established.*

   The data is collected in the way specified by the command-line options, for which see Command-Line Tool for Export to ServiceNow. To prevent timeout issues, each data set is split into segments for transfer to ServiceNow. Each segment is identified with a transaction ID, transaction type, and the data in an XML chunk. ServiceNow returns a return code for each segment of data.

   - If the return code shows a failure, the segment is retransmitted for a maximum number of tries (the default value of 10 can be changed in the configuration file), after which the utility exits (skipping any remaining exports) and returns `1` to the command line.
   - While success continues, the utility waits for each data type to complete before commencing the next data type.
5. In ServiceNow, as the first data chunk is received, the integration application creates a record in the `Import Runs` table. (For details of the main columns in the `Import Runs` table, see Import Runs Columns.)

   Note: *If you have selected to store installed application data to SAM Foundation Plugin tables but the SAM plugin is not found, then ServiceNow logs an error in the Application Log and returns an error code. In FlexNet Manager Suite, the utility abandons the export and displays an appropriate error in the log file.*

6. For each data chunk (including the first), ServiceNow creates a record in the `Import Transactions` table. (For details of the main columns in the `Import Transactions` table, see Import Transactions Columns.)
7. When the transaction record is created, it returns a success code to FlexNet Manager Suite, which then processes and transmits the next data chunk. This loop continues until all the records of a particular export type have been transmitted.
8. In ServiceNow, the inbound data chunks are written to staging tables (visible in the navigation panel under **FlexNet Manager Suite > Staging Records**) based on the transaction type:

Transaction type	Import Set Table
`inventory_export`	`Inventory Imports` (note that this is hardware inventory)
`application_export`	`Application Imports`
`contract_export`	`Contract Imports`
`connection_test`	Not applicable
`export_status`	Not applicable

9. In ServiceNow, the integration application uses the `Scheduled Data Imports` and `Data Sources` records to process each `Import Transaction` record. The integration application queues the transaction records, and as each record is processed, it sets its `State` to `Succeeded`. When all the transaction records for an `Import Run` are completed, the integration application also sets its `State` to `Succeeded`.

10. During this processing loop, FlexNet Manager Suite continues polling the API for status. Only when it receives a successful completion message does it resume the process with the next selected data type.

11. When the data has been collated from the individual transactions to the staging tables, transform maps are executed to map the fields in the import set tables to fields in ServiceNow database tables. The final tables for data from each transaction type are shown below. (For a complete set of the transform mappings, see Transform Maps for ServiceNow Integration.)

Transaction type	Ultimate Data Tables in ServiceNow
`application_export`	`Software Model`, `Discovery Model`, `Software Installation`, and `Software Instance`. The tables used are dependent upon the **Use CMDB and/or SAM tables for installation** integration property. For more information about integration properties, refer to Integration Properties.
`contract_export`	`Contract` and `Lease Instance` ***Tip:*** *Check the* ***Contract used by*** *tab near the bottom of the page of contract properties. This references all computers linked to the contract.*
`inventory_export`	`Product Model`, `Virtual Machine Instance`, `Computers`

At the completion of this process, data exported from FlexNet Manager Suite is reflected in your ServiceNow data set.

Import Runs Columns

These are the major properties of the entries in the **FlexNet Manager Suite > Integration Runs > Import Runs** view created as ServiceNow commences each data import.

Column	Notes
`Number`	An automatically-generated sequential numbering of each created record in this listing.
`Import Id`	A GUID created by FlexNet Manager Suite as it exports the data, used to track activity and exchanges for this particular import into ServiceNow.
`Import Type`	The type of data being imported, being one of `Inventory`, `Contract`, or `Application`.
`State`	Values include: • `New` — This default state is set when a record is created in the **Import Runs** listing. • `Waiting` — ServiceNow has received the final data chunk for the particular `Import Type`. • `Succeeded` — All data chunks for this `Import Run` have been processed by ServiceNow. • `Failed` — For some reason (unspecified here), the import has failed.

Import Transactions Columns

These are the major properties of the entries in the **FlexNet Manager Suite > Integration Runs > Import Transactions** view created as ServiceNow receives each data segment for import.

Column	Notes
`Number`	An automatically-generated sequential numbering of each created record in this listing.
`Import Run`	The ID of the import run of which this transaction is a part.
`Import Type`	The type of data being imported, being one of `Inventory`, `Contract`, or `Application`.

Column	Notes
State	Values include: • New — This default state is set when a record is created in the **Import Transactions** listing. • Waiting — ServiceNow is processing another transaction received earlier. • Processing — Set when the integration application invokes import of this transaction. • Succeeded — ServiceNow has completed the import of this transaction. • Failed — For some reason (unspecified here), the import of the transaction has failed.
Payload	The segment of data, in XML format, that makes up this transaction.

Transform Maps for ServiceNow Integration

These transform maps map hardware, license, and contract data collected from FlexNet Manager Suite into ServiceNow.

Data collected from FlexNet Manager Suite is initially held in staging tables within ServiceNow, and must be transformed for insertion in the operational tables in your ServiceNow implementation. For example, the `FNMS Inventory -> Computer` set has columns that are matched with the `Computers` table in ServiceNow. Contract data is first mapped into the `Contracts` table in ServiceNow, and a second transform map is run to provide links to relevant `Configuration Items` (Computers).

The following tables show the standard transformations from the staging tables (the source) to operational tables (the target) within ServiceNow. Items marked `[Script]` involve a data transformation, which may involve finding the foreign key to another record, or conversion of units such as bytes to MB.

In each transform, the "(key)" fields are used for record matching. If a record already exists with identical key values for fields so marked, it is updated; and if not, a new record is created.

Note: *There are integration properties that define:*

- *Whether a new record should be created in ServiceNow when a match for incoming data is not found*
- *Whether an existing record should be updated when a match is found*

These properties can be modified in ServiceNow by navigating to ***FlexNet Manager Suite > Integration Properties****. For more information, see Integration Properties.*

The transform maps in this topic are separated into the following respective data group that they are used for:

- Transform maps for inventory data
- Transform maps for installed application data
- Transform maps for contracts data

Transform maps for inventory data

Computer Model Transform

- Original data from FlexNet Manager Suite: Inventory
- Target table in ServiceNow: `Product Model [cmdb_model]`

Source Display Name	Source Field	Target Display Name	Target Field
ModelNo (key)	u_modelno	Model number	model_number
ChassisType	u_chassistype	Type	type
ComputerType	u_computertype	Model categories	cmdb_model_category
Manufacturer	u_manufacturer	Manufacturer	manufacturer
ModelNo	u_modelno	Name	name

Virtual Machine Transform

- Original data from FlexNet Manager Suite: Inventory.
- Target table in ServiceNow: `Virtual Machine Instance [cmdb_ci_vm_instance]`

Source Display Name	Source Field	Target Display Name	Target Field
ComputerID (key)	u_computerid	FlexNet Computer ID	x_fls_flexera_fnms_computer_id
[Script]	[Script]	Is deleted	x_fls_flexera_fnms_is_deleted
[Script]	[Script]	Correlation ID	correlation_id
[Script]	[Script]	Class	sys_class_name
[Script]	[Script]	Disks size (GB)	disks_size
[Script]	[Script]	Memory (MB)	memory
ComputerName	u_computername	Name	name
Domain	u_domain	Domain	sys_domain
IPAdress	u_ipaddress	IP Address	ip_address
MACAddress	u_macaddress	MAC Address	mac_address
Manufacturer	u_manufacturer	Manufacturer	manufacturer
ModelNo	u_modelno	Model ID	model_id
ModelNo	u_modelno	Model number	model_number
NumberOfProcessors	u_numberofprocessors	CPUs	cpus
SerialNo	u_serialno	Serial number	serial_number

Computer Transform

- Original data from FlexNet Manager Suite: Inventory
- Target table in ServiceNow: `Computer [cmdb_ci_computer]`

Source Display Name	Source Field	Target Display Name	Target Field
ComputerID (key)	u_computerid	FlexNet Computer ID	x_fls_flexera_fnms_computer_id
[Script]	[Script]	RAM (MB)	ram
[Script]	[Script]	Correlation ID	correlation_id
[Script]	[Script]	Disk space (GB)	disk_space
AssetID	u_assetid	FlexNet Asset ID	x_fls_flexera_fnms_asset_id
CalculatedUser	u_calculateduser	Calculated User	x_fls_flexera_fnms_calculated_user
ChassisType	u_chassistype	Chassis type	chassis_type
ComputerName	u_computername	Name	name
ComputerStatus	u_computerstatus	Status (hardware_status)	hardware_status
ComputerType	u_computertype	Subcategory	subcategory
DiscoveredDate	u_discovereddate	Discovered date	x_fls_flexera_fnms_discovered_date
Domain	u_domain	Domain	sys_domain
InventoryConnection Name	u_inventoryconnection name	Inventory Connection	x_fls_flexera_fnms_inventory_connection
InventorySource	u_inventorysource	Inventory Source	x_fls_flexera_fnms_inventory_source
IPAddress	u_ipaddress	IP Address	ip_address
IsDeleted	u_isdeleted	Is deleted	x_fls_flexera_fnms_isdeleted
LastLoggedInUser	u_lastloggedinuser	Last logged in user	x_fls_flexera_fnms_last_logged_in_user
MACAddress	u_macaddress	MAC Address	mac_address
Manufacturer	u_manufacturer	Manufacturer	manufacturer
MaxClockSpeed	u_maxclockspeed	CPU speed (MHz)	cpu_speed
ModelNo	u_modelno	Model ID -> model_number	model_id
ModelNo	u_modelno	Model number	model_number
NumberOfCores	u_numberofcores	CPU core count	cpu_core_count
NumberOfProcessors	u_numberofprocessors	CPU count	cpu_count
NumberOfThreads	u_numberofthreads	CPU core thread	cpu_core_thread
OperatingSystem	u_operatingsystem	Operating System	os

Source Display Name	Source Field	Target Display Name	Target Field
ProcessorType	u_processortype	CPU type	cpu_type
SerialNo	u_serialno	Serial number	serial_number

Transform maps for installed application data

Software Transform

- Used when you opt to use CMDB tables for Installed Application data.
- Original data from FlexNet Manager Suite: Application.
- Target table in ServiceNow: `Software [cmdb_ci_spkg]`

Source Display Name	Source Field	Target Display Name	Target Field
FlexeraID (key)	u_flexeraid	Flexera Unique ID	x_fls_flexera_fnms_id
ApplicationID	u_applicationid	FlexNet Application ID	x_fls_flexera_fnms_application_id
ApplicationName	u_applicationname	Package name	package_name
ApplicationVersion	u_applicationversion	Version	version
Classification	u_classification	Subcategory	subcategory
ProductName	u_productname	Name	name
Publisher	u_publisher	Manufacturer	manufacturer

Software Instance Transform

- Used when you opt to use CMDB tables for Installed Application data.
- Original data from FlexNet Manager Suite: Application.
- Target table in ServiceNow: `Software Instance [cmdb_software_instance]`

Tip: *When application data is imported into ServiceNow, the installed application records are linked to their computer record based on `FlexNet Computer ID`. This is recommended and the most reliable way to link records. However, if for any reason you need to link records based on `Computer Serial Number`; note that the installed application records do have `Computer Serial Number` available and therefore you can manually edit the transform map in ServiceNow to change the key to be `Computer Serial Number`.*

Source Display Name	Source Field	Target Display Name	Target Field
FlexeraID (key)	u_flexeraid	Software -> Product Name	Product Name
[Script] (key) [Note 1]	[Script]	Installed on	installed_on
[Script]	[Script]	Is deleted	x_fls_flexera_fnms_is_deleted

Source Display Name	Source Field	Target Display Name	Target Field
DiscoveredAt	u_discoveredat	Discovered at	x_fls_flexera_fnms_discovered_at
DiscoveredBy	u_discoveredby	Discovered by	x_fls_flexera_fnms_discovered_by
FlexeraID	u_flexeraid	Flexera Unique ID	x_fls_flexera_fnms_id
LastScanned	u_lastscanned	Last scanned	x_fls_flexera_fnms_last_scanned
ProductName	u_productname	Name	name

Note:

1. An installed application record defines the installation of an application on a computer. A source record from FlexNet Manager Suite in the `Application Import [x_fls_flexera_fnms_application_import]` table has these values.
   The import process creates or updates such a record in ServiceNow in the `Software Instance [cmdb_software_instance]` table. The record links an application record in the `Software [cmdb_ci_spkg]` table and a computer record in `Computer [cmdb_ci_computer]`.

   Field map, [Script] (key), finds a computer record in the `Computer` table based on source `ComputerID [u_computerid]` in the target `FlexNet Computer ID [x_fls_flexera_fnms_computer_id]` field. Another field map, `FlexeraID (key)` finds an application record in the `Software` table based on the source `FlexeraID [u_flexeraid]` field in the target `Flexera Unique ID [x_fls_flexera_fnms_id]` field.

Software Model Transform

- Used when you opt to use SAM tables for Installed Application data.
- Original data from FlexNet Manager Suite: Application
- Target table in ServiceNow: `Software Model [cmdb_software_product_model]`

Source Display Name	Source Field	Target Display Name	Target Field
FlexeraID (key)	u_flexeraid	FlexNet Manager Id	x_fls_flexera_fnms_id
ApplicationName	u_applicationname	FlexNet Application Name	x_fls_flexera_fnms_application_name
ApplicationVersion	u_applicationversion	Version	version
Classification	u_classification	Type	type
FlexeraID	u_flexeraid	Short description	short_description
ProductName	u_productname	Name	name
Publisher	u_publisher	Manufacturer	manufacturer

Software Installation Transform

- Used when you opt to use SAM tables for Installed Application data.
- Original data from FlexNet Manager Suite: Application

- Target table in ServiceNow: `label [cmdb_sam_sw_install]`

Tip: *When application data is imported into ServiceNow, the installed application records are linked to their computer record based on `FlexNet Computer ID`. This is recommended and the most reliable way to link records. However, if for any reason you need to link records based on `Computer Serial Number`; note that the installed application records do have `Computer Serial Number` available and therefore you can manually edit the transform map in ServiceNow to change the key to be `Computer Serial Number`.*

Source Display Name	Source Field	Target Display Name	Target Field
ApplicationID (key)	u_applicationid	FlexNet Application ID	x_fls_flexera_application_id
[Script] (key) [Note 1]	[Script]	Installed on	installed_on
[Script]	[Script]	Is deleted	x_fls_flexera_fnms_is_deleted
ApplicationVersion	u_applicationversion	Version	version
DiscoveredAt	u_discoveredat	Last Discovered	x_fls_flexera_fnms_last_discovered
DiscoveredBy	u_discoveredby	Discovered by	x_fls_flexera_fnms_discovered_by
DisplayName	u_displayname	Display name	display_name
FlexeraID	u_flexeraid	Discovery model -> prod_id	discovery_model
FlexeraID	u_flexeraid	Prod id	prod_id
LastScanned	u_lastscanned	Last scanned	last_scanned
Publisher	u_publisher	Publisher	publisher

Note:

1. An installed application record defines the installation of an application on a computer. A source record from FlexNet Manager Suite in the `Application Import [x_fls_flexera_fnms_application_import]` table has these values.
   The import process creates or updates such a record in ServiceNow in the `Software Installation [cmdb_sam_sw_install]` table. The record links an application record in the `Software Discovery Model [cmdb_sam_sw_discovery_model]` table and a computer record in `Computer [cmdb_ci_computer]`.

   Field map, [Script] (key), finds a computer record in `Computer` table based on source `ComputerID [u_computerid]` in the target `FlexNet Computer ID [x_fls_flexera_fnms_computer_id]` field. Another field map, `FlexeraID (key)` finds an application record in the `Software Discovery Model` table based on the source `FlexeraID [u_flexeraid]` field in the target `Prod ID [prod_id]` field.

Transform maps for contract data

Contracts Transform

- Original data from FlexNet Manager Suite: Contracts
- Target table in ServiceNow: `Contract [ast_contract]`

Source Display Name	Source Field	Target Display Name	Target Field
ContractNumber (key)	u_contractnumber	Contract number	vendor_contract
ContractName	u_contractname	Description	short_description
ContractStatus	u_contractstatus	State	state
ContractType	u_contracttype	Short Description	short_description
EndDate	u_enddate	Ends	ends
IsDeleted	u_isdeleted	Is deleted	x_fls_flexera_fnms_is_deleted
StartDate	u_startdate	Starts	starts
Vendor	u_vendor	Vendor	vendor

Contract Instance Transform

- Original data from FlexNet Manager Suite: Contracts
- Target table in ServiceNow: `Lease Instance [ast_contract_instance]`

Source Display Name	Source Field	Target Display Name	Target Field
ContractNumber (key)	u_contractnumber	Contract	ast_contract -> vendor_contract
ContractType	u_contracttype	Contract Type	contract_type
See note.	See note.	Configuration Item	ci_item

Note: *Script is run to query `cmdb_ci_computer` table. If the CI is found, the script returns the `sys_id` for REF. If the CI is not found, this field is left blank.*

Command-Line Tool for Export to ServiceNow

This tool extracts data from the FlexNet Manager Suite operations database, and imports it into ServiceNow.

Note: *This utility may be triggered in three different ways:*

- *Automatically by a scheduled task*
- *Manually by an operator through the web interface for FlexNet Manager Suite. Navigate to the system menu (in the top right corner)* ***System Settings > Integrations****.*
- *From the command line, as described in this topic.*

In all cases, it exports the data specified to ServiceNow. However, when it is run manually from the command line, the behavior of the utility then differs: it does not update the results in the web interface for FlexNet Manager Suite. For this reason, automatic operations are the preferred method of operation; but this command line is available for testing and troubleshooting.

Syntax:

```
fnmp_servicenow_export.exe
                -url ServiceNow-URL
                -user ServiceNow-account-name
                -token ServiceNow-token
                -endpoint ServiceNow-endpoint
                -changes Boolean
                -changesPeriod -numberOfDays
                -changesSince YYYY-MM-DD
                -connectionTest true
                -withApplications Boolean
                -withContracts Boolean
                -WithHardwareInventory Boolean
```

Option	Sample value	Notes
-url	`https://YOUR-INSTANCE.service-now.com`	Mandatory. The URL to your ServiceNow server instance in the cloud. Use the HTTPS protocol.
-user	`account@mydomain.com`	Mandatory. This is the user name specified in the appropriate credentials set within ServiceNow, the same credentials from which the token is collected.
-token	86 random alpha-numeric characters	Mandatory. This is the refresh token copied from ServiceNow.
-endpoint	`fnmp.do` *or* `api/x_fls_flexera_fnms/integration/fnmstosn`	Mandatory. Leave the value unchanged. This is the integration point for ServiceNow. ***Tip:*** *While the endpoint is visible in ServiceNow and could be modified there, it is also recorded in the operations database for FlexNet Manager Suite. Changing it in ServiceNow without a matching change in the operations database will cause automated exports to fail.*

Option	Sample value	Notes
-changes	`true`	Optional, with a default of `true` if it is not present. When this option is not present, or when it is present with a value of `true` (case insensitive), only differential changes are included in all data exports to ServiceNow. For the baseline from which differences are measured, see the **-changesSince** or **-changesPeriod** options. When this option is present and has a value `false`, all data (from the relevant period) is exported from the relevant tables in FlexNet Manager Suite (which is likely to make the data transforms in ServiceNow considerably more time-consuming). ***Tip:*** *You may use the default value even for an initial data transfer, or for the first transfer after an upgrade from an earlier version of the executable. Initial transfers are always the full data set, with differential transfers following thereafter.*
-changesPeriod	`-5`	Optional (no effect when not present). When present, specifies the time window (as a number of days) before the export for which changes are included in the export. The negative sign is mandatory. The example means that changes in the previous five days are exported. Days are measured as complete 24-hour blocks prior to the time of the export.
-changesSince	`2015-02-28`	Optional. Use only when necessary to over-ride the default date, which is the last run of the batch scheduler on the central batch server for FlexNet Manager Suite (or, for smaller on-premises implementations, the server hosting this functionality). The format is `YYYY-MM-DD`.
-connectionTest	`true`	Optional. If this option is provided, and has a value `true` (case insensitive), no data exports are attempted. The executable attempts a connection to ServiceNow, and reports the results (`0` or `2` as described below) on the console. In order to transfer data from FlexNet Manager Suite to ServiceNow, this parameter must either be omitted, or set to `false`.

Option	Sample value	Notes
-withApplications	`false`	Optional. When omitted, it is assumed to be `true`, and insatalled applications are included in the export. If set to `false`, installed applications are excluded from the export. Note that at least one of these `-with*` parameters must be true.
-withContracts	`false`	Optional. When omitted, it is assumed to be `true`, and contracts are included in the export. If set to `false`, contracts are excluded from the export. Note that at least one of these `-with*` parameters must be true.
-WithHardware Inventory	`false`	Optional. When omitted, it is assumed to be `true`, and hardware devices are included in the export. If set to `false`, hardware devices are excluded from the export. Note that at least one of these `-with*` parameters must be true. ***Caution:*** *In ServiceNow, contracts and installed applications both have dependencies on hardware inventory. It is recommended that you do not exclude the hardware inventory, as this may result in unpredictable gaps in other records when the correct dependencies cannot be established.*

Exit Codes

`0`	Success
1	A data upload to ServiceNow has failed through the maximum number of retries set in the configuration file.
2	Connection failure, being either of: • Failed to connect to ServiceNow. Check the URL and account details for connection. • Failure to connect to the FlexNet Manager Suite compliance database.
3	There is no valid license for execution of the export utility recorded for your enterprise.
4	Incomplete set of command-line parameters provided (see syntax listing above). Help information is displayed in this case.
5	An invalid `tenantUId` was provided. (Not applicable to on-premises implementations.)
6	Export from FlexNet Manager Suite cannot proceed because ServiceNow has not completed previous pending imports.

Example: Examples

This example runs a connection test (because of the final parameter). Although the defaults are set for all data transfers (installed applications, contracts, and hardware inventory), no exports occur. The command-line result is `0` for good credentials, and `2` for bad credentials.

```
fnmp_servicenow_export.exe
       -token
DbBNKvekyBVGFxPXe7TAGooeyb5bsByB2jSubap232lAkerucGHn5SnlaQasincR89...
       -user myname@mycorp.com
       -url https://myserver.service-now.com
       -endpoint fnmp.do
       -connectionTest true
```

The following example restricts the export to collect only hardware inventory. Because it is not mentioned, the default true value includes hardware inventory, with the other two possibilities excluded. Because the `-changes` and `-changesSince` options are both missing, only hardware changes occurring since the last scheduled export are included.

```
fnmp_servicenow_export.exe
       -token
DbBNKvekyBVGFxPXe7TAGooeyb5bsByB2jSubap232lAkerucGHn5SnlaQasincR89...
       -user myname@mycorp.com
       -url https://myserver.service-now.com
       -endpoint fnmp.do
       -withContracts false
       -withApplications false
```

This example covers all export types from FlexNet Manager Suite, but including only changes that occurred in the previous 3 days, measured back in 24-hour blocks from the time the export is run. Assuming this example is run at 1pm Thursday, it will export only changes from 1pm Monday to 1pm Thursday (all times are local on the batch server). Strictly, `-changes true` is redundant, since this is the default.

```
fnmp_servicenow_export.exe
       -token
DbBNKvekyBVGFxPXe7TAGooeyb5bsByB2jSubap232lAkerucGHn5SnlaQasincR89...
       -user myname@mycorp.com
       -url https://myserver.service-now.com
       -endpoint fnmp.do
       -changes true
       -changesPeriod -3
```

Process for Exports from ServiceNow to FlexNet Manager Suite

How the import of data from ServiceNow works.

Process Details

1. Two scheduled jobs are run separately (and not at the same time) in ServiceNow.

   Note: *These jobs are visible in* ***System Definition*** > ***Scheduled Jobs*** *as* `Export Assets from ServiceNow` *and* `Export Contracts from ServiceNow`*. Click the job names to make them active or inactive, to modify the schedule, or to execute either one immediately.*

2. When either job is executed, ServiceNow creates an XML file on the MID server with the data from the database view (**FlexNet Manager Suite > Database Views**). The filename is `x_fls_flexera_fnms_asset.xml` for assets or `x_fls_flexera_fnms_contract.xml` for contracts. The files are overwritten with every export.

   Tip: *Within the database views (****FlexNet Manager Suite > Database Views****), you can specify additional tables to join to the database view, or within a table, you can add additional columns to join to the database view. To preview the records for an export (without actually invoking the export), click the view name in that page to display the properties of the view; and under the* ***Related Links*** *heading, click* `Try It`*.*

3. After the export is executed in ServiceNow, a record is created under **FlexNet Manager Suite > Integration History > Export Runs**. ServiceNow updates the state of this record to `Succeeded` when the export is successfully completed.

4. Two independent business adapters from FlexNet Manager Suite (on potentially independent schedules) connect from your inventory beacon to your MID server, and collect the latest XML files saved there, converting them into the intermediate data form required for business data uploads. (For details of the mapping between source data in ServiceNow and destination fields in the FlexNet Manager Suite database, see Business Adapter Mappings.)

   Important: *The business adapters will fail if they are run while the XML files on the MID server are incomplete. Be sure that the schedules for export from ServiceNow, and the running of the business adapters, allow sufficient time for the export to be completed. For large data sets, this may be several hours.*

5. As the adapters complete their run, the data set is uploaded immediately to the application server for FlexNet Manager Suite. (There is also a catch-up upload task that runs overnight to retry any failed uploads.)

6. On a separate schedule, the batch server starts a business import job. This imports data from the data package into the FlexNet Manager Suite database.

At the completion of this process, data exported from ServiceNow is reflected in your FlexNet Manager Suite data set.

Properties for Export Columns

These are the major properties of the entries in the **FlexNet Manager Suite > Integration History > Export Runs** view created as ServiceNow commences each data export.

Column	Notes
`Number`	An automatically-generated sequential numbering of each created record in this listing.
`Export Type`	The type of data being exported, either `Contract` or `Asset`.

Column	Notes
`Stage`	May have the following values: • `File Creation` — Displayed when the integration application sends the opening XML tag to the MID Server to be commence writing the XML file. • `Data Collection` — Displayed while the integration application gathers the data and sends it to MID Server to be written into the XML file. The data is sent in multiple chunks of 500 records each. • `File Completion` — Displayed when the closing XML tag has been sent to the MID Server.
`State`	Provides additional insight into the `Data Collection` stage in particular. Values include: • `New` — This default state is set when a record is created in the **Export Runs** listing. • `Processing` — The integration application is reading from the ServiceNow database and compiling a chunk of data. • `Processed` — All database records required for the current chunk (maximum: 500) have been prepared. • `Waiting` — The data chunk is being transferred to the MID server. • `Ready` — The MID server has notified that the data chunk has been added to the XML file. At this point, the integration application sets the state back to `Processing` and prepares the next chunk of data. • `Succeeded` — All data chunks have been transferred, and no more are required. • `Failed` — For some reason (unspecified here), the export has failed.
`Counter`	The number of database records that the integration application has read and processed. Because the data is transferred to the MID server in a number of chunks, the counter value is used as a cursor for reading the correct records from the database for the next data chunk.

Business Adapter Mappings

Two business adapters are supplied as a standard part of the ServiceNow integration application for FlexNet Manager Suite, and are used in the process of transferred asset and contract data from ServiceNow to FlexNet Manager Suite. The following lists show the source data from ServiceNow, and the target (destination) fields in FlexNet Manager Suite, for each of these adapters in turn.

Tip: *For special purposes, mappings can be added, removed, or changed by editing the business adapter(s) in Business Adapter Studio on your inventory beacon.*

In each listing, the first field shown is used for matching existing records (a key). During import, if the value in this field already exists in the compliance database, the record is updated with the other values imported for this item. Otherwise, a new record is automatically created in FlexNet Manager Suite.

In both listings below, "display" names are the names visible in the two products' web interfaces; and the **Source Field** column shows a compound name made up of:

- An abbreviated reference to a ServiceNow database table name (see below)
- An underscore character
- The field name in the relevant database table (which name may contain additional underscores).

The mapping of the abbreviated references to the actual database table names in the ServiceNow database is:

- `acntrct` represents `ast_contract`
- `ahrdwr` represents `alm_hardware`
- `cicomp` represents `cmdb_ci_computer`
- `cmdbmdl` represents `cmdb_model`
- `sysusr` represents `sys_user`.

Assets Imports

- Business adapter file name: `ServiceNowAssets.xml`
- Original data source: ServiceNow database view `x_fls_flexera_fnms_asset`
- Destination table in FlexNet Manager Suite database: `Asset`

Source Display Name	Source Field	Target Display Name	Target Field
Serial number	`cicomp_serial_number`	**Serial Number**	`SerialNumber` (key)
Asset tag	`ahrdwr_ asset_tag`	**Asset Tag**	`AssetTag`
Asset Status	`ahrdwr_install_status`	**Asset Status**	`AssetStatus`
Configuration Item	`ahrdwr_ci`	**Name**	`ShortDescription`
Installed	`cicomp_install_date`	**Installed on**	`InstallationDate`
Manufacturer	`cicomp_manufacturer`	**Manufacturer**	`Manufacturer`
ModelNo	`cicomp_model_number`	**ModelNo**	`ModelNo`

Notes:

1. `AssetStatusID` is a foreign key to the `AssetStatus` table, from which display values are drawn.

Note: *`AssetStatus` is a mandatory field for asset records in FlexNet Manager Suite. If a record in ServiceNow has a* ***Status*** *value that does not exist in FlexNet Manager Suite:*

- *The new status value is inserted into the `AssetStatus` table (you may also need to customize localized values of*

this status, if other languages are in use within your enterprise)

- *An `AssetStatus` is automatically created for this new entry*
- *The `AssetStatus` is inserted into the asset record.*

If you are 'round-tripping' this data (that is, copying it back from FlexNet Manager Suite to ServiceNow at some future point, to maintain synchronization), the new value is returned when appropriate.

Contracts Imports

- Business adapter file name: `ServiceNowContracts.xml`
- Original data source: ServiceNow database view `x_fls_flexera_fnms_contract`
- Destination table in FlexNet Manager Suite database: `Contract`

Source Display Name	Source Field	Target Display Name	Target Field
Contract number	`acntrct_vendor_ contract`	**Contract Number**	`ContractNo` (key)
Description	`acntrct_description`	**Information**	`Comments`
Display name	`cmdbmdl_display_name`	**Contract type**	`ContractTypeID` [Note 1]
Ends	`acntrct_ends`	**Expiry date**	`EndDate`
Short description	`acntrct_short_description`	**Contract Name**	`ContractName`
Starts	`acntrct_starts`	**Start date**	`StartDate`
State	`acntrct_state`	**Status**	`ContractStatusID` [Note 2]

Notes:

1. `ContractTypeID` is a foreign key to the `ContractType` table, from which display values are drawn. Also see note below.
2. `ContractStatusID` is a foreign key to the `ContractStatus` table, from which display values are drawn.

Note: *`ContractTypeID` is a mandatory field for contract records in FlexNet Manager Suite. If a record in ServiceNow has a **Display name** value that does not exist in FlexNet Manager Suite:*

- *The new contract type is inserted into the `ContractType` table (you may also need to customize localized values of this type, if other languages are in use within your enterprise)*
- *An `ContractTypeID` is automatically created for this new entry*
- *The `ContractTypeID` is inserted into the contract record.*

If you are 'round-tripping' this data (that is, copying it back from FlexNet Manager Suite to ServiceNow at some future point, to maintain synchronization), the new value is returned when appropriate.

5

Appendices

The following topics cover removal of any previously-installed legacy integration application (any version less than 4.0), and improving the performance of ServiceNow with additional database indexes. A possible exception in the ServiceNow log files is also explained.

Integration Properties

The **Integration Properties** page lets you specify export integration properties. This page is accessible from **FlexNet Manager Suite > Integration Properties**. The integration properties let you choose whether to accept FlexNet Manager Suite as the source of truth for various data types. Choosing **No** for any option sets ServiceNow as the source of truth for that property and choosing **Yes** for any option sets FlexNet Manager Suite as the source of truth for that property.

The descriptions in the following sections show you which properties are for use with exports from ServiceNow and which properties are for use with exports from FlexNet Manager Suite.

Properties used when exporting from ServiceNow for input to FlexNet Manager Suite

The following table provides descriptions of properties to use when exporting from ServiceNow for input to FlexNet Manager Suite.

Setting/Property	Description
Exclude virtual machine assets	Choose whether to exclude virtual machine assets from the export from ServiceNow.
MID server name	Enter the full name of the MID server.
Export file path	Optionally, enter a file path for where to store the exported XML data file from ServiceNow (instead of storing it in the default MID server location). Because the FlexNet Beacon reads this file using the business adapter, FlexNet Beacon needs access to this file location.

Setting/Property	Description
Configure the log verbosity (View the Application logs at System Logs - System Log - Application logs)	Choose the desired level of log verbosity: • **Error** • **Warning** • **Information** • **Debugging** • **Tracing**

Properties used when exporting from FlexNet Manager Suite for input to ServiceNow

The following table provides descriptions of properties to use when exporting from FlexNet Manager Suite for input to ServiceNow.

Setting/Property	Description
FlexNet Manager Suite is source of truth for:	Choose **Yes** to accept FlexNet Manager Suite as the source of truth for any of the following: • **Adding inventories** • **Updating inventories** • **Updating inventory class name (sys_class_name)** • **Adding installations** • **Updating installations** • **Adding contracts** • **Updating contracts** Choose **No** to any of the above settings to specify ServiceNow as the source of truth for that setting.
Use CMDB and/or SAM tables for installations	Choose what tables in ServiceNow that you want imported data to go in: • **Decide automatically**—Import to SAM tables. If no SAM tables exist, data is instead imported to CMDB tables. • **Use CMDB tables**—Import to CMDB tables. • **Use SAM tables**—Import to SAM tables. If there are no SAM tables, the import will fail. • **Use CMDB and SAM tables**—Import to CMBD tables **and** SAM tables. If SAM tables do not exist, the import will fail.

Setting/Property	Description
Configure the log verbosity (View the Application logs at System Logs - System Log - Application logs)	Choose the desired level of log verbosity: • **Error** • **Warning** • **Information** • **Debugging** • **Tracing**

Additional ServiceNow Indexes for Performance

Creating additional indexes on your ServiceNow instance substantially improves the intake of data exported from FlexNet Manager Suite.

ServiceNow enables you to create your own database indexes. A worked example suggests that, with the following indexes added, you can expect better than four-fold performance increase in the initial data transforms to finish the import of data that has been exported from FlexNet Manager Suite.

Database table	New column indexed
`Contract [ast_contract]`	`vendor_contract`
`Software Model [cmdb_software_product_model]`	`x_fls_flexera_fnms_id`
`Product Model [cmdb_model]`	`model_number`
`Computer [cmdb_ci_computer]`	`x_fls_flexera_fnms_computer_id`
`Software Installation [cmdb_sam_sw_install]`	`x_fls_flexera_fnms_application_id`

Removing a Legacy Integration Application

The integration application for ServiceNow (from version 3.0 and later of the adapter) is a scoped application. The previous integration applications used global scope. While two integration applications can be present in ServiceNow at the same time (with different scopes), it is recommended that you remove any previously-installed integration application for FlexNet Manager Suite after installing the newer version and successfully running the migration script.

Note: *Using the following process to remove all the components of the legacy integration application (including its menu, tables, scripts, and user interface components). As expected, this process does not remove any records from ServiceNow core tables.*

To remove a legacy integration application:

1. In ServiceNow, navigate to **System Applications > Applications**, and click **FlexNet Manager Suite Integration**.
2. On the page that appears, click **Delete**.

   A confirmation dialog appears asking you to type in the word “delete”.
3. Enter `delete` in the dialog, and click **Ok**.

   The application deletion progress dialog appears, and the integration application is deleted.

   Tip: *Sometimes deleting the integration application fails to remove the Scheduled Jobs it created. To check, or to delete them manually:*

   a. In ServiceNow, navigate to **System Scheduler > Scheduled Jobs > Scheduled Jobs**.

   b. Filter the list of scheduled jobs to find those with `Name` starting with `Export`.

   c. If the following scheduled jobs exist, delete them manually:

   - `Export Assets from ServiceNow`
   - `Export Contracts from ServiceNow`.

Configuring the Software Asset Management Foundation Plugin

When you export application data from FlexNet Manager Suite for use in ServiceNow, the export utility `fnmp_servicenow_export.exe` makes use of tables that are part of a plugin for ServiceNow. The naming of this plugin, and your method of enabling it, have changed considerably across various releases of ServiceNow. Choose the appropriate step below depending on your current version of ServiceNow.

Summary of procedures across ServiceNow releases:

1. Up to and including the Helsinki release of ServiceNow, the following applies:
   - The plugin name is `Software Asset Management`, and its ID is `com.snc.software_asset_management`.
   - This plugin has a dependency on a second one named Software Asset Management Core with the ID `com.snc.sam.core`.
   - Your ServiceNow administrator can make the `Software Asset Management` plugin `Active` in the **System Plugins** page of your ServiceNow instance. This action automatically makes the Software Asset Management Core plugin active as well.
2. For the Istanbul release of ServiceNow, the following applies:
   - If you had activated the `Software Asset Management` plugin in an earlier version of ServiceNow, and your ServiceNow instance has now been upgraded to Istanbul, there should be no need to activate the plugin a second time.

- The plugin feature is called "Software Asset Management Template", but still accessed through the **System Plugins** page of your ServiceNow instance.
- The plugin name is `Software Asset Management`, and its ID is `com.snc.software_asset_management`.
- (There is no visible dependency on any other plugin, as had been the case for earlier releases of ServiceNow.)
- Your ServiceNow administrator can make the `Software Asset Management` plugin `Active` in the **System Plugins** page of your ServiceNow instance.

3. For the Jakarta release of ServiceNow, the following applies:

- If you had activated the `Software Asset Management` plugin in an earlier version of ServiceNow, and your ServiceNow instance has now been upgraded to Jakarta, there should be no need to activate the plugin a second time.
- Confusingly, there is a new and different plugin named `Software Asset Management Premium`, and its ID is `com.snc.samp`. This is used internally by a new SAM application that is subscription based.

  Important: *This new subscription-based application and its plugin are not relevant to the integration with FlexNet Manager Suite, and this subscription is not required for the integration. Do not be confused by the similarities. Be aware that, when current ServiceNow documents refer to "Software Asset Management" or "SAM", they are generally referring to the new application and the premium plugin it uses, and this is not relevant to your project for integration with FlexNet Manager Suite.*

- The plugin from previous releases, named `Software Asset Management` or `Software Asset Management Template`, with its ID `com.snc.software_asset_management`, is no longer visible in the **System Plugins** page of your ServiceNow instance. However, it is still supported for the Jakarta release, and is still required for the integration between ServiceNow and FlexNet Manager Suite.

  Tip: *Some documents from ServiceNow refer to this plugin as the "Base SAM plugin" or the "old SAM plugin", and the like. Be prepared to be precise, and refer to its ID* `com.snc.software_asset_management` *to be sure the correct plugin is identified.*

- To activate the "base SAM plugin", send a request to ServiceNow Support to activate the `Software Asset Management Template` plugin *with the ID* `com.snc.software_asset_management` on your ServiceNow instance.

4. For the Kingston release of ServiceNow, the following applies:

- For the Kingston release, do *not* activate the earlier plugin named `Software Asset Management`, with its ID `com.snc.software_asset_management`. This legacy plugin is only applicable to the Jakarta and earlier releases.
- Instead, for Kingston, activate the `Software Asset Management Foundation` plugin, with its ID `com.snc.sams`. This plugin does not require any additional licensing.
- Contact ServiceNow Support to request activation of the `Software Asset Management Foundation` plugin (`com.snc.sams`).

Deleting Records from ServiceNow

If an asset or application record exists both in FlexNet Manager Suite and ServiceNow but is then deleted in FlexNet Manager Suite, the records are marked for deletion in ServiceNow during the next export from FlexNet Manager Suite. However, the records are not automatically deleted. To delete the records from ServiceNow, you need to create a Global business rule and include a script with a condition: `current.state == 'Succeeded'`.

To delete records in ServiceNow using a global business rule:

1. In ServiceNow, navigate to **System Definition**, and click **Business Rule**.

   The **Business Rules** page appears.

2. Click **New**.

3. Do the following.

   a. In the **Name** field, enter a name for the rule:

   b. From the **Table** field, select **x_fls_flexera_fnms_import_transaction**.

   c. Ensure that the **Application** field is set to **Global**.

   d. Select the **Active** check box.

   e. Select the **Advanced** check box.

   f. On the **Advanced** tab, enter the following in the **Condition** field:

   ```
   current.state == 'Succeeded'
   ```

   g. On the **Advanced** tab, enter the following into the **Script** field:

   ```
   var util = new FNMSUtil();

   util.removeIsDeletedRecordsFromTable('cmdb_ci_computer');

   util.removeIsDeletedRecordsFromTable('cmdb_software_instance');

   util.removeIsDeletedRecordsFromTable('cmdb_ci_vm_instance');

   util.removeIsDeletedRecordsFromTable('cmdb_sam_sw_install');
   ```

   Note: *The last line (`util.removeIsDeletedRecordsFromTable('cmdb_sam_sw_install');`) should only be included if your setup is using the SAM Foundation plugin.*

XIII

Tanium Connector

This part introduces the Tanium Connector, provided by Flexera, which allows you to export inventory data from Tanium into FlexNet Manager Suite. Tanium Asset collects a complete inventory of your hardware and software assets for both online and offline devices, including servers, laptops, and desktops.

FlexNet Manager Suite uses the software and hardware inventory data collected from Tanium Asset to calculate license consumption. As always, the inventory records must be recognized by the Application Recognition Library, and you must have the resulting application records linked to the appropriate license, for compliance calculations to proceed.

Tanium Connector must be installed on FlexNet Manager Suite as outlined in the Installing Tanium Connector on FlexNet Manager Suite prerequisite.

Supported versions

The Tanium connector supports inventory import from the following releases of Tanium:

- Tanium Server version 7.3.0 or later
- Tanium Asset version 1.6.3 or later.

1

Prerequisites

The following topics provide the software prerequisites that are required and the configuration steps that must be undertaken ***before*** you can create a connection to Tanium.

- FlexNet Beacon Server Prerequisites
- Tanium Server Prerequisites
  - Creating a Flexera SQL Edition view within Tanium Asset
- Installing Tanium Connector on FlexNet Manager Suite

FlexNet Beacon Server Prerequisites

FlexNet Beacon Software Prerequisites

The following FlexNet Beacon server prerequisites are required for the Tanium Connector:

- FlexNet Beacon 2019 R2 (14.0) or later which is shipped with FlexNet Manager Suite 2019 R2 or later
- PowerShell 5.1 or later
- A web browser that can access FlexNet Manager Suite
- The ability to login to the inventory beacon, and run FlexNet Beacon, using an account with administrator privileges.

Tanium Server Prerequisites

Tanium Server Software Prerequisites

The following Tanium server software prerequisites are required to use the Tanium Connector:

- An operational Tanium instance
- Tanium Server version 7.3 or later

- Tanium Asset version 1.6.3 or later
- A web browser
- The ability to login to Tanium web interface, using an account with administrator privileges.

Note: *Tanium Connector leverages the REST APIs provided by Tanium to export asset inventory data from Tanium Asset and import it into FlexNet Manager Suite.*

Tanium Server Configuration Prerequisites

1. Complete the following three configuration tutorials which are documented in the Tanium Asset User Guide on the https://docs.tanium.com/asset/asset/exportasset.html?Highlight=flexera page:
   a. *"Configure the Flexera staging database"*

      Tip: *You must configure your staging database to allow a Flexera destination in Tanium. Although Tanium Connector will not use the staging database to import inventory data, the staging database is required to add a Flexera destination in Tanium.*

   b. *"Add Flexera destination"*
   c. *"(Optional) Enable file evidence content"*
2. Once the above configurations are complete, you are ready to create a Flexera SQL Edition view. See Creating a Flexera SQL Edition view within Tanium Asset.

Creating a Flexera SQL Edition view within Tanium Asset

You must manually create the Flexera SQL Edition view in Tanium Asset to enable Tanium Connector to correctly recognize Microsoft SQL Server and retrieve these evidences from Tanium and import the evidence data into FlexNet Manager Suite.

1. Login to Tanium Asset.
2. Expand the left-hand menu and navigate to **Views**.

   The list of existing destinations displays.
3. Click **New View**.

   The **Create View** page displays.
4. In the **View Name** field, enter `Flexera SQL Edition`.

   This exact name must be entered.
5. In the **Description** field, you can enter any description.
6. In the **View Attributes** section, click the **Select None** link to deselected all of the check boxes and then select the following attributes:

- **Computer Name**
- The following attributes which are in the **Asset Internal** section:
  - **Asset Id**
  - **Created At**
  - **Updated At**
- The following attributes which are in the **Asset SQL Server Details** section:
  - **SQLServerEdition**
  - **SQLProductVersion**
  - **SQLProductLevel**
  - **SQLDisplayName**
  - **SQLInstanceName**

7. Click **Create View**.

   The Flexera SQL Edition view displays in the list of Asset Views.

Installing Tanium Connector on FlexNet Manager Suite

How to install Tanium Connector on FlexNet Manager Suite.

Is Tanium Connector already installed?

The Tanium Connector may already be in place in your FlexNet Manager Suite implementation. You can check this by inspecting any inventory beacon as follows:

1. If the inventory beacon is already running, close and then reopen in as an administrator. This ensures the Tanium Connector package is downloaded to your inventory beacon server.
2. Open Windows Explorer and navigate to your `<drive>\<path>\ProgramData\Flexera Software\Compliance\ImportProcedures\Inventory\Reader`, for example the default location is `C:\ProgramData\Flexera Software\Compliance\ImportProcedures\Inventory\Reader`.
   - If a `Tanium` folder exists, this means the Tanium Connector is already in place and you are ready to Configure FlexNet Beacon to connect with Tanium.
   - If the `Tanium` folder does not exist, then you must complete the following steps to install the Tanium Connector.

Installing Tanium Connector

1. Use your browser to access the Flexera Customer Community.
   a. On https://flexeracommunity.force.com/customer/CCLanding, use the account details emailed to you with your order confirmation from Flexera to log in (using the **Login** link in the top right).

> ***Tip:*** *Access requires your Customer Community user name and password. If you do not have one, use the* `Request Community Access` *link on the login page to request one. Your credentials are configured for access to content you have licensed.*

b. Select the **Downloads** tab from the row across the top of the page.

A routing page appears to let you `Access Product and License Center`, displaying lists of products from Flexera.

c. In the lists of products, identify `FlexNet Manager Platform`, and immediately below it, click **Access Above Products**.

The `Product and License Center` site is displayed.

d. In the `Your Downloads` section of the `Home` page, click the link for FlexNet Manager Platform.

e. In the `Download Packages` page, click the link for FlexNet Manager Platform 2019 R2 to access the downloads. (You may need to repeat this action on a second page to access the downloadable files.)

2. Click the `Adapter Tools for` FlexNet Manager Suite `Update 2019R2-01.zip`link and download to a convenient location (such as `C:\temp`).

This archive file contains the required TaniumConnector.zip file.

3. To install the Tanium Connector on FlexNet Manager Suite:

a. On the FlexNet Manager Suite central application server (or the batch server in a multi-server implementation), extract the downloaded TaniumConnector.zip file to a temporary local folder.

b. Navigate through the unzipped archive to `TaniumReader` folder and follow the steps in the `readme` file.

4. To update the FlexNet Beacon:

a. On the FlexNet Beacon server, extract the downloaded TaniumConnector.zip file to a temporary local folder.

b. Navigate through the unzipped file until you see the `Beacon` folder and follow the steps in the `readme` file.

5. To replace your entire FlexNet Manager Suite online help so it includes the Tanium Connector topics:

a. On the web application server, extract the downloaded TaniumConnector.zip file onto a local folder.

b. Navigate through the unzipped archive to the `WebHelp` folder and follow the steps in the `readme` file.

2

Configure FlexNet Beacon to connect with Tanium

How to configure FlexNet Beacon to connect with Tanium.

1. Log into the inventory beacon interface as an administrator (for example, in the Windows Start menu, search for **FlexNet Beacon**, right-click it, and select **Run as administrator**).

   Tip: *Remember that logging into an inventory beacon requires an account with administrator privileges.*

2. Select the **Inventory systems** page and click the down arrow on the right of the **New...** split button, and choose **PowerShell**.

   Tip: *You can also edit a connection you have defined previously, by selecting it from the list of connections and clicking* ***Edit...****.*

3. In the dialog that appears, complete (or modify) the following required fields:

   - **Connection Name**: The name you give this inventory connection is also used in the web interface of FlexNet Manager Suite to name the data import task. The name may contain alphanumeric characters, underscores or spaces, but it must start with either a letter or a number, for example `Tanium`.
   - **Source Type**: Select **Tanium** from the drop-down list.

4. Optional, if your enterprise uses a proxy server to enable communication between FlexNet Beacon and Tanium, select the **Use Proxy** check box, and complete the following additional details:

   a. **Proxy Server**: Enter the address of the proxy server as an IP address (for example `IPAddress:PortNumberIPAddress:PortNumber`)), or a HTTP address (for example `http://ProxyServerURL:PortNumber`).

      If the port number is omitted, the default port is `:80` and if the protocol is omitted the default is `http:`.

   b. **Username** and **Password**: If your enterprise uses an authenticated proxy, specify the credentials to access the proxy server you just identified.

5. In the **Tanium** section, complete the following fields:

   a. **Tanium Server**: Enter the IP address of your Tanium instance, for example 10.20.300.40

      Do not include the http:// or https:// protocol with the IP address.

   b. **Username** and **Password**: Enter the username and password for your Tanium instance.

6. Click **Test connection** to validate that you have specified the details correctly and adjust as necessary.

   When FlexNet Beacon has successfully accessed the Tanium APIs using the details supplied, a `Test connection succeeded` message displays.

7. Click **OK** and then click **Save** to close this dialog .

   The new connection is added to the connections list in the **Inventory Systems** page.

8. If you do not already have a suitable schedule for this connection available on this inventory beacon, select the **Scheduling** page and click **New...** to open the **Edit Schedule** dialog and define one.

   For more details see *Scheduling a Connection* in the online help.

9. Back on the **Inventory systems** page, select the new connection in the list, click **Schedule...** to select your appropriate schedule in the **Select schedule** dialog and then click **OK** to confirm your selection.

10. In the main **FlexNet Beacon** interface, click **Save** to store your newly scheduled connection.

11. Select the newly added connection and click **Execute Now**.

    The **Execute Now** confirmation dialog displays and the import of Tanium inventory data commences.

12. Click **Ok** to confirm.

The resulting import (which follows immediately) may take some time to complete. You may monitor progress in the web interface for FlexNet Manager Suite — navigate to the system menu (⚙ ▼ in the top right corner) and choose **System Health > System Tasks**, and periodically refresh this page. When the process is complete, you can inspect the inventory imported from Tanium in the **All Inventory** page on FlexNet Manager Suite.

Tip: *In the **All Inventory** page:*

1. *Add the **Connection name** column to the page.*
2. *Display the simple filter bar.*
3. *Filter for the name you have your Tanium connection (the example given was `Tanium`).*

After a successful data import, the hardware, software inventory data from Tanium Asset are visible in the appropriate pages of FlexNet Manager Suite.

Note: *To know more about the operations available on the **Inventory Systems** page of FlexNet Beacon, see Inventory Systems Page in the online help. For scheduling data imports through this connection, see Scheduling a Connection, also in online help.*

3

Validating Imports to the Inventory Beacon

How to validate imports to the inventory beacon.

This process traces the path of data imported from Tanium through an inventory beacon, to the central application server of FlexNet Manager Suite. You may start this process at whichever stage is suitable to your needs. To validate imports to the inventory beacon:

1. On the inventory beacon that connects to the Tanium Connector, log into FlexNet Beacon using an account with administrator privileges.
2. On the **Inventory Systems** page, select your connection to Tanium, and click **Execute Now**.

   A confirmation shows that the extract has started.
3. In the top-left corner of the FlexNet Beacon interface, right-click the menu icon, and select **Open log file folder**.

   File Explorer opens.
4. Ensure that you are viewing `ProgramData\Flexera Software\Compliance\Logging\` and then navigate to the `ComplianceReader` folder.
5. In the text editor of your choice, open the text document importer-[*nnnn*] that was modified on today's date, and scroll to the bottom to determine whether data reading is still complete. If not, exit the text editor, and try again in a few minutes.

   A successful process shows the following at the end of the log file (with each line prefixed with the date and time):

   ```
   [INFO] 0 source data warnings
   ```

   ```
   [INFO] 0 errors, 0 warnings
   ```

   ```
   [INFO] Import has been completed successfully
   ```

   If there are warnings or errors in the log file, try to correct the issues. If you are unable to correct the error, save the log file and attach it to a support ticket. Typical issues at this stage may include:

   - Proxy or firewall settings preventing access to the Internet

- Intermittent issues with the Tanium Server
- Incorrect details for the account name or password accessing Tanium
- The connection has not been regularly scheduled and so succeeds only when executed manually.

Note: *Immediately after a successful import is completed, the inventory beacon attempts to upload the archive of imported data to its parent (the parent may be another inventory beacon in the hierarchy, or the application server):*

- *When the upload succeeds, the inventory beacon saves a copy in your default drive at* `<Drive>:\ProgramData\Flexera Software\Beacon\IntermediateData\Uploaded` *and preserves this copy for 14 days*
- *If the upload fails, the imported archive stays in* `<Drive>:\ProgramData\Flexera Software\Beacon\IntermediateData`.

Tip: *If you are working on a disconnected inventory beacon (one that cannot access the application server), this* `IntermediateData` *folder contains the data files that you must transfer to another location where uploads are possible.*
You may now want to review or debug the imported data. To review the data imported from Tanium:

a. *Navigate to* `<Drive>:\ProgramData\Flexera Software\Beacon\IntermediateData\Uploaded`*, and inspect the most recent relevant .zip file there. If the zip archive remains in the* `IntermediateData` *folder (on a connected inventory beacon), there is an upload problem, as discussed below.*

To debug uploads from this inventory beacon to its parent:

- *Review* `C:\ProgramData\Flexera Software\Compliance\Logging\ComplianceUpload\upload.log` *for details of any networking issues and note the URL of the next server in the upload chain. If this is another inventory beacon, repeat these reviews there, and so on until you exhaust the hierarchy of inventory beacons.*

The upload URL is in a log line that starts with `dateTime [Upload.UploadProcess ] [INFO] Uploading to http`

If you are unable to resolve problems, collect your logs together in a zip archive. Ask your registered support contact (a designated person within your enterprise who has access rights and login details) to open a new support case at https://community.flexera.com/t5/forums/postpage/board-id/@support, including a clear description of the issue.

4

Tanium Connector Architecture

This chapter provides a useful framework for your understanding of the more detailed workflow of the Tanium Connector.

The above diagram shows the workflow of how the connector works:

Tanium

- Creating Flexera Destination will create the views and reports automatically
- Flexera views are used for importing data from Tanium Asset to FlexNet Manager Suite Imported* tables, later Compliance database tables.
- The following Tanium views used for importing data:
  - Flexera Computer — contains Computer, User, Domain and WMI evidences data
  - Flexera Software Installations — contains Installer evidences data
  - Flexera File Evidence — contains File evidences data
  - Flexera Network Adapter — contains IP address and mac addresses data for each computer
  - Flexera SQL Edition — contains SQL related installer evidence data.

> ***Note:*** *You must create this view manually, if the view does not exist. See* *Creating a Flexera SQL Edition view within Tanium Asset* *for instructions.*

Tip: *Flexera reports in Tanium Asset enable crosschecking of the data between Tanium and FlexNet Manager Suite once the import is complete.*

Beacon

- In Beacon Server, connects to Tanium to perform the inventory upload
- Data will be collected using these views through REST API calls and creates intermediate zip file.

FlexNet Manager Suite

- Intermediate zip file will be pushed from Beacon to FlexNet Manager Suite using Writers logic (data into the Compliance database).
- Once data is imported, this inventory is matched to the Application Recognition Library and license compliance is calculated.

5

Investigating Tanium Connection Errors

You can troubleshoot FlexNet Beacon failures or errors by reviewing the error messages that display in the FlexNet Beacon web interface, or in the log files for any failures, errors or warning messages. If you are unable to resolve problems, collect your logs together in a zip archive. Ask your registered support contact (a designated person within your enterprise who has access rights and login details) to open a new support case at https://community.flexera.com/t5/forums/postpage/board-id/@support, including a clear description of the issue. The following table provides help with potential issues that you may encounter when attempting to connect to Tanium.

Error, Failure or Message	Description
`Value is required`	One or more mandatory fields are missing a value in the PowerShell Source Connection dialog when configuring FlexNet Beacon. You can resolve this error by ensuring all mandatory field are completed and retesting and save the connection. For details, see Configure FlexNet Beacon to connect with Tanium.
`View missing: Flexera SQL Edition`	If the Flexera SQL Edition view is missing, this message displays when you test the connection between in the FlexNet Beacon and the Tanium server. You can resolve this error by creating the Flexera SQL Edition view. See Creating a Flexera SQL Edition view within Tanium Asset.
`The remote name could not be resolved: '<[fully qualified domain name of Tanium server] of the Tanium server>'`	This message is most likely to occur because there is no Internet connection available to the location where the inventory beacon is installed. It may also occur for the following reasons: • The remote name could not be resolved • The Tanium server is blocked by a firewall or server is currently down • The fully qualified domain name of the Tanium server and the IP address have not been added in the host file.

Error, Failure or Message	Description
`Test connection failed`	This message displays in the PowerShell Source Connection dialog if the Flexera Destination is deleted from the Tanium server. You can resolve this error by adding the Flexera destination. See https://docs.tanium.com/asset/asset/exportasset.html?Highlight=flexera for instructions.
`Unable to connect to the remote server`	This message may appear when http://or https:// is included as part of the IP address to the Tanium Server. You can resolve this error by removing the http://or https:// from the **Tanium Server** field. For details, see Configure FlexNet Beacon to connect with Tanium.

6

Appendix: Technical Data

The following information is not required for set-up or operation of the Tanium Connector.

The connection to Tanium lists imported hardware and software inventory. Details of imported data can be viewed on the **All Inventory** page by filtering results by the **Connection name** you choose. You can then select the **Name** to view full list of properties that have been imported for a single inventory device.

Imported inventory data displays in the **General**, **Hardware**, **Applications**, and **Evidence** tabs as appropriate. See online help for full details.

Hardware inventory data Imported by Tanium Connector

An inventory device record is automatically created for your hardware inventory (with the resulting property sheet). The following properties (listed alphabetically) are collected for hardware inventory through the Tanium Connector.

Tanium View Name	Columns	FNMS (Table Name)/Column Name
Flexera Computer	**Id**	
	asset_ad_short_domain	ImportedDomain /FlatName
	asset_last_logged_in_user	ImportedComputer/LastLoggedOnUser ImportedUser/SAMAccountName ImportedUser/Domain
	asset_number_of_cpu_socket	ImportedComputer/NumberOfSockets
	chassis_type	ImportedComputer/ChassisType
	computer_name	ImportedComputer/ComputerName
	cpu_core	ImportedComputer/NumberOfCores
	cpu_name	ImportedComputer/ProcessorType
	cpu_processor	ImportedComputer/NumberOfProcessors
	cpu_speed	ImportedComputer/MaxClockSpeed
	disk_total_space	ImportedComputer/TotalDiskSpace

	display_adapter_count	ImportedComputer/ NumberOfDisplayAdapters
	domain_name	ImportedDomain/QualifiedName
	manufacturer	ImportedComputer/Manufacturer
	model	ImportedComputer/ModelNo
	number_of_fixed_drive	ImportedComputer/NumberOfHardDrives
	number_of_logical_processor	ImportedComputer/ NumberOfLogicalProcessors
	operating_system	ImportedComputer/OperatingSystem
	os_platform	ImportedWMIEvidence /ClassName
	serial_number	ImportedComputer/SerialNo
	service_pack	ImportedComputer/ServicePack
	system_uuid	ImportedComputer/UUID
	ram	ImportedComputer/TotalMemory
	updated_at	ImportedComputer/ ServicesInventoryDate
Flexera Network Adapter	**Id**	
	ipv4_address	ImportedComputer/IPAddress
	mac_address	ImportedComputer/MACAddress
Flexera File Evidence	**Id**	
	company	ImportedFileEvidence/Company
	description	ImportedFileEvidence/Description
	file_name	ImportedFileEvidence/FileName
	file_path	ImportedFileEvidence/FilePath
	file_size	ImportedFileEvidence/FileSize
	file_version	ImportedFileEvidence/FileVersion
	language	ImportedFileEvidence/Language
	product_name	ImportedFileEvidence/ProductName
	product_version	ImportedFileEvidence/ProductVersion
Flexera Software Installations	**Id**	
	ci_installed_application_name	ImportedInstallerEvidence/DisplayName
	ci_installed_application_source	ImportedInstallerEvidence/Evidence

	ci_installed_application_version	ImportedInstallerEvidence/Version
	ci_installed_application_vendor	ImportedInstallerEvidence/Publisher
Flexera SQL Edition	**Id**	
	ci_asset_sql_server_sqldisplayname	ImportedInstallerEvidence/DisplayName
	ci_asset_sql_server_sqlproductversion	ImportedInstallerEvidence/Version
	ci_asset_sql_server_sqlserveredition	ImportedInstallerEvidence/DisplayName

XIV

XenApp Server Adapter

The XenApp server adapter, provided by Flexera, allows you to collect software inventory from Citrix Virtual Apps (formerly Citrix XenApp) and import it into FlexNet Manager Suite. Depending on the type of virtualized applications being served, the evidence appears in either the installer evidence list (for App-V or streaming profile applications delivered through Citrix Virtual Apps), or in the file evidence list (for file-based applications managed through Citrix Virtual Apps).

In either case, the evidence must be linked to an application record. This can happen in either of two ways:

- Where the evidence is matched by an existing inventory rule for an application, it is automatically linked to that application record.
- Where required details (below) are incomplete, or there is no exact match in any existing evidence rules/records (either supplied by the Application Recognition Library or created locally in your enterprise), the evidence is left in the **Discovered Evidence** page (and **All Evidence** page) with its **Assigned** property set to No. The fields that require matching are:
  - For installer evidence, the application name, version and publisher
  - For file evidence, the file name, version, company, and description.

Once the evidence is linked to an application, the application must be linked to a license. The license should then be linked to purchase records to determine your entitlements. Of course, these are manual tasks outside the scope of the adapter's operations.

The term *XenApp server* is used in this documentation as a generic term to cover the differently-named control servers for different versions of Citrix Virtual Apps:

- In version 6.x, the XenApp server was officially named the Zone and Data Collector. One such controlling server was required per *farm*.
- In version 7.5 and later, the XenApp server is called the Delivery Controller. One such controlling server is required per *delivery site*.

Supported versions

Citrix Virtual Apps server (previously XenApp server) 6.0, 6.5, 7.5-7.9, 7.11–7.17 (XenApp), 7.1808–7.1909 (Citrix Virtual Apps)

If it happens that you have multiple of these versions of Citrix Virtual Apps in operation (for example in different domains),

you can use the same structure described in this section to link them all to FlexNet Manager Suite.

1

Architecture, Operations and Prerequisites

This chapter provides a useful framework for your understanding of the more detailed content to follow.

Architecture and Operation

In order to track licenses for applications delivered remotely to users from a Citrix Virtual Desktops App environment, FlexNet Manager Suite needs information about which users and devices have access to which applications. There are several sources of such data available from XenApp servers, depending on the version of Citrix Virtual Desktops (formerly XenApp):

- For XenApp 6.0 and 6.5:
  - Access control lists (ACLs)
  - Streaming profiles
  - Citrix EdgeSight servers.
- For XenApp 7.5 and later:
  - Access control lists (ACLs)
  - User filters collected from Applications and Delivery Groups
  - App-V 5 packages (and the applications they contain).

  Tip: *No usage tracking is possible for XenApp 7.5, as in this release Citrix did not include usage tracking capabilities, in XenApp.*

- For XenApp 7.6 and later:
  - The XenDesktop database supplied as part of XenApp that tracks application usage.
- For XenApp 7.9 and later:
  - The collection of Application Group data

- User filters collected from Application Groups.

  Note: *A one time staging database update is required to use XenApp server agent to collect Application Group data and to collect user filters from Applications, Application Groups, and Delivery Groups.*

- For Citrix Virtual Apps 7.1808 and later:
  - The collection of Application Group data
  - User filters collected from Application Groups.

  Note: *A one time staging database update is required to use XenApp server agent to collect Application Group data and to collect user filters from Applications, Application Groups, and Delivery Groups.*

These sources are discussed in turn in the following sections.

Access control lists (ACLs)

These are lists which specify the permissions associated with an object, such as an application's executable file, on a server. The FlexNet Manager Agent for XenApp Server (XenApp server agent) is a tool that extracts information about users and which applications they can access remotely, and transfers that information to an inventory beacon for use in licensing calculations in FlexNet Manager Suite. To do this, the XenApp server agent must be installed:

- For Citrix XenApp 7.5 (and later) and Citrix Virtual Apps 7.1808 (and later), on one Delivery Controller for each Delivery Site. If you have multiple Delivery Sites, you may choose either of the following:
  - Install the XenApp server agent on one Delivery Controller in each Delivery Site
  - Use only a single XenApp server agent, and provide that XenApp server agent with the required network access and credentials to access all required Citrix Virtual Apps Delivery Controllers.
- For XenApp 6.0 or 6.5, on one controlling XenApp server in each Citrix farm.

Tip: *While the XenApp server agent is installed only on one server per farm, for XenApp 6.x you also need software inventory from every XenApp server, in order to identify the editions of applications available to users and computers. This separate inventory of the XenApp servers can be obtained either by installing the FlexNet inventory agent on the XenApp server, or using Zero-footprint inventory collected by an inventory beacon. Do not get the two separate agents (FlexNet inventory agent, and XenApp server agent) confused. The main focus of this adapter documentation is the XenApp server agent.*

The XenApp server agent is supplied as an integral part of the XenApp server adapter.

Tip: *In earlier releases, the XenApp server agent extracted Active Directory names and details of users and devices from the XenApp server. Now, the XenApp server agent collects only Active Directory SIDs (a large performance improvement). As a result, best practice is that your inventory beacon completes its import of Active Directory data before importing Citrix Virtual Apps data, so that all Active Directory SIDs can be resolved against the user names, devices, and groups collected directly from Active Directory.*

Streaming profiles for XenApp version 6.0 and 6.5

The XenApp server agent is also able to read the contents of the `.profile` and the key executable files associated with

streamed applications published to your XenApp servers.

Tip: *As the streaming profile is not stored in the XenApp server's database, the XenApp server agent must have at least read access to the streaming profile location to be able to read and extract this information.*

As these applications are not physically installed on your XenApp server, combining the XenApp server agent's data from `.profile` files with EdgeSite server information may be the only way for FlexNet Manager Suite to recognize usage of such applications.

Note: *FlexNet Manager Suite is only able to recognize usage of files streamed to a XenApp server, not those streamed directly to client devices.*

Citrix EdgeSight servers for XenApp 6.0 and 6.5

Citrix EdgeSight for XenApp monitors and profiles the usage of remote and streamed applications by users, telling you both who is using that application, and on what device. The data from EdgeSight is very valuable for FlexNet Manager Suite: you may use it for license optimization (for example, tightening access through ACL permissions to exclude users who evidently do not need to use the applications); or it may be critical for any user-based or usage-based licensing of applications delivered through XenApp 6.0 or 6.5.

EdgeSight agents may be installed on each XenApp server, and report back to a central EdgeSight server, which can keep track of application usage on multiple XenApp machines, belonging to one or more farms. The FlexNet Beacon can connect to each EdgeSight server and collect this usage information for use in compliance calculations.

Unlike data from the XenApp server agent, EdgeSight data does contain details of which devices access a particular application. Thus, EdgeSight data is usually more valuable to an enterprise for calculating license compliance than the data about application availability returned by the XenApp server agent alone. If, however, your enterprise deploys streamed applications, EdgeSight usage data may need to be supplemented by XenApp server agent information to accurately recognize these applications.

The following information is returned from the EdgeSight server:

- A list of applications (product name, version, publisher, and description) and the users who use them
- The devices on which users request and run applications
- The XenApp servers from which users request applications
- The farms to which the XenApp servers belong.

Tip: *The EdgeSight data does not include application editions. Because different XenApp servers may have different editions installed (and available to users), it is important to take software (and hardware) inventory of the XenApp servers themselves, using the FlexNet inventory agent (either installed locally on each server or operating remotely from an inventory beacon). This inventory reveals the software editions available on each of the servers, which can be combined with the information listed above to give complete usage data required for license calculations. For example, if server XenApp01 offers Visio Standard, while server XenApp02 offers Visio Professional, the inventory from XenApp01 and XenApp02, combined with the data listed above, allows the license consumption calculations to link users to the appropriate license.*

To use EdgeSight data, you must create a database connection to the EdgeSight SQL server database.

App-V 5 packages for Citrix XenApp 7.5 (and later) and Citrix Virtual Apps 7.1808 (and later)

The XenApp server agent is able to inspect the contents of App-V 5 packages and recover the name, version, and publisher of the application contained in each package. The XenApp server agent also returns the user's ability to *access* these App-V packages (as recorded in the ACLs described earlier). However, in the ability to track which users actually *use* the applications, there are differences across versions:

- Version 7.5 has no technology like the EdgeSight server available in version 6.x, and so cannot report application usage
- From version 7.6, XenApp (Citrix Virtual Apps) again allows tracking application usage through connection to the Citrix Virtual Desktops (formerly XenDesktop) database incorporated in XenApp 7.6 and later.

VDI images for Citrix XenApp 7.5 (and later) and Citrix Virtual Apps 7.1808 (and later)

The same capabilities apply to VDI images. The XenApp server agent interrogates any VDI device managed by the XenApp server to read the applications listed in all VDI master images available (including spinning up any images that are currently dormant to inspect their applications). As with App-V packages:

- For XenApp version 7.5, there is no ability to track who uses any VDI image, or when
- For Citrix XenApp 7.6 (and later) and Citrix Virtual Apps 7.1808 (and later), the included Citrix Virtual Desktops (formerly XenDesktop) database allows collection of application usage information.

Changed architecture across versions

Because XenApp release 7.5 follows an extensive rewrite of the XenApp line by Citrix, the architectures of the two systems (and therefore the ways that the adapter integrates with the architecture) are quite different from version 6.x to 7.x.

Tip: *The following three diagrams do not include the import of Active Directory data by the inventory beacon, as this is not part of the adapter itself. However, the prior import of Active Directory data (typically, by the same inventory beacon connecting to the staging database) is a prerequisite for operation of the adapter.*
Also keep clearly in mind the distinction between the XenApp server agent, an executable installed on your XenApp server(s), and the XenApp server adapter(s), an XML file (with embedded SQL) installed on your inventory beacon. The XenApp server agent is responsible for the first part of the process, collecting data and normally writing it into a staging database; and the XenApp server adapter(s) takes the second part, loading data from the staging database to the central application server and its compliance database.

This diagram represents the architecture and data flows for the XenApp server adapter(s) connected to XenApp 6.0 or 6.5 (the key numbers are referenced in the table below):

The next diagram shows the architecture and data flows when connected to XenApp 7.5. The diagram is laid out similarly to highlight the changes in architecture, and in particular the absence of usage information:

Finally, the third diagram shows the architecture and data flows when connected to Citrix XenApp 7.6 (or later) and

Citrix Virtual Apps 7.1808 (or later). The main point to note is the return of usage information:

The following table shows the data collected by the adapter through the different channels numbered in the three diagrams.

Table 4: Lists imported by XenApp server adapter

List	Case 1	Case 2	Case 3	Case 4
User SIDs, Active Directory Group SIDs	Y		Y	
File evidence (`.exe`) details – file name, version, company, description	Y	Y	Y	
Installer evidence (from App-V/streaming profiles) details – name, version, publisher	Y	Y	Y	Y
Application access rights per user SID	Y		Y	
Application usage per user		Y		Y*
Client computer SIDs (with user SIDs)		Y		Y
XenApp servers with applications present for delivery	Y			
App-V packages (and the applications therein) managed by Citrix Virtual Apps (formerly XenApp)			Y	
XenApp servers that served applications		Y		
Applications available in App-V packages (creates App-V 'evidence' records too)			Y	Y

List	Case 1	Case 2	Case 3	Case 4
Applications available as streaming profiles	Y			

* Application usage by users and devices for Citrix XenApp 7.6 (and later) and Citrix Virtual Apps 7.1808 (and later) releases is limited to applications delivered in App-V packages. (Usage based on imported file evidence is not collected.)

Operation without a staging database

It is possible to vary the integration architecture (shown in previous diagrams) so that it does not require a separate staging database. This is achieved by using additional command-line options for the XenApp server agent that allow it to connect directly to an inventory beacon (see XenApp Server Agent Command Line Options for details). The inventory beacon uploads the collected data through the central application server (or, in a large multi-server implementation, the inventory server) directly into pre-staging tables in your compliance database.

This model still requires the XenApp server adapter installed on the inventory beacon, and you must still configure a connection to the "incoming" data (as described in Create Connections for Data Upload) — except that in this case, instead of connecting to an external staging database, the connection is directly to your central database server. The XenApp server adapter continues to perform its regular work on the central database, de-duplicating, normalizing and transforming data from the pre-staging tables in the compliance database into the appropriate `Imported*` staging tables in the same database, where it awaits the next compliance import.

This alternative varies the third diagram above to look more like this:

Operation with Citrix XenApp 7.5 (and later) or Citrix Virtual Apps 7.1808 (or later)

The XenApp server agent is installed on the XenApp server (at your discretion, on only one server that can access all

other controllers for Citrix Virtual Apps (formerly XenApp) in your enterprise, or as many as one per Citrix Virtual Apps site).

Triggered by a Windows scheduled task, the agent runs according to your settings. It may collect inventory details only from the XenApp server on which it is installed (default), or it may collect from several controlling XenApp servers in sequence (identified with the `-s` command line option, detailed in XenApp Server Agent Command Line Options).

a. VDI images

Based on information found on the XenApp server, the XenApp server agent may connect to any relevant XenApp servers hosting VDI images that contain applications. XenApp allows an administrator to nominate applications within VDI images for delivery

1. As individual applications only
2. Within a VDI (delivered as a whole environment) only
3. Either as individual applications or within a VDI.

The XenApp server agent collects information on all applications within the VDI master images that are identified for individual delivery (options 1 or 3 above). The VDI evidence is returned to FlexNet Manager Suite as file evidence. (For XenApp version 7.6 and later, usage tracking is not available for the file evidence.)

Tip: *To track inventory delivered within an entire VDI environment (option 2 above), use XenDesktop discovery and inventory through the rules-based process in the web interface of FlexNet Manager Suite.*

b. App-V packages

Again based on information gathered from the XenApp server, the XenApp server agent connects to any Microsoft App-V publication server to inspect any App-V packages registered for delivery through XenApp. Because XenApp requires App-V version 5 (or later) for integration, the XenApp server agent can interrogate the packages to identify the applications inside. The App-V evidence is returned to FlexNet Manager Suite as installer evidence.

For XenApp version 7.6 and later (but not for version 7.5), a connection is also made to the XenDesktop database from the appropriate inventory beacon. This collects details of App-V application packages that users and devices have accessed. This additional information restores the ability (missing for version 7.5) to determine which users and devices have actually used each application, as distinct from merely having access to them. This may allow for more accurate license consumption calculations in later compliance calculations for applications delivered in this way.

c. Processing

So both kinds of applications are returned to FlexNet Manager Suite as evidence:

- From VDI images, file evidence is produced that normally includes file name, version, company, and description
- For App-V packages, installer evidence is returned that normally includes application name, version, and publisher.

When the data is finally imported into FlexNet Manager Suite:

- The incoming evidence is tested against existing evidence records (including "rules" generalized with wild cards) already linked to applications (either from the Application Recognition Library or from records produced in your enterprise).
- If the incoming evidence matches any existing rule or record, it is recorded against the linked application, and its presence is recorded in the properties of the appropriate user or device as an "installation" record. For XenApp 7.6 (or later), a usage record is automatically created for each user and each device shown to have accessed the

application.

- If the incoming evidence is not matched, it is displayed in evidence listings (for example, navigate to **License Compliance > Discovered Evidence** (in the **Evidence** group), selecting the **Installer evidence** tab for App-V applications and the **File evidence** tab for VDI applications). You can select the evidence in the appropriate tab, and click **Assign** to choose an application record (or create a new one) to link to the evidence. (You only need do this the first time that new evidence is reported. Once linked to the application, your evidence serves as a 'rule' for matching future imports of the same evidence.)
- Once the incoming evidence is linked to an application (either automatically or manually), license reconciliation attempts to calculate consumption through XenApp on any license linked to the application. For this to take effect, you must correctly configure at least one license attached to the application:
  1. Navigate to the **Use rights & rules** tab of the license properties.
  2. Ensure that the **License consumption rules** heading is expanded (if not, click the heading).
  3. Select **Access granted to users, or usage, consumes license entitlements** to expose additional controls.
  4. Depending on the terms of your license, choose one of **Consume one entitlement for each user** or **Consume one entitlement per device owned by each user**.
  5. For XenApp 7.5, set **Consume entitlements based on** to `Access` (because only access records are available through XenApp 7.5). For other versions tracking App-V applications, check the terms of your license to see whether usage-based licensing is acceptable for this application, and make selections accordingly.

Collection of Application Group data and user filters from Applications, Application Groups, and Delivery Groups

The collection of Application Group data from XenApp server and XenApp Desktop is supported in FlexNet Manager Suite for XenApp and XenDesktop versions 7.9 and later. FlexNet Manager Suite also supports the collection of user filters from Applications, Application Groups, and Delivery Groups to more accurately calculate application access. (XenApp and XenDesktop provide the ability to add user filters to Applications, Application Groups, and Delivery Groups in order to restrict access.)

Note: *To take advantage of support for Application Groups and the collection of user filters, you must use the new XenApp server agent,* ***XenAppAgent****. In addition, a one time staging database update is required to use* ***XenAppAgent*** *to collect Application Group data and user filters from Application Groups. The* ***XenAppAgent*** *installer can be found in the within the* ***XenAppAgent*** *folder of the* ***Citrix XenApp Server Agent*** *subdirectory that is provided in the* ***Adapter Tools for FlexNet Manager Suite*** *archive. The* ***Adapter Tools for FlexNet Manager Suite*** *archive is available in a zip archive in the Product and License Center.*

Tip: *Application Groups are groups that let you manage collections of applications. Being able to categorize applications into collections enables some settings to be administered for all applications in a group. The concept of Application Groups was added in the 7.9 release of XenApp and XenDesktop.*

Prerequisites

The XenApp server adapter requires the following:

- The executable and supporting files for the XenApp server agent. These are available as described in Creating the Staging Database.
- The XenApp server (on which the XenApp server agent is installed) requires:
  - .NET version 4.5 or greater
  - PowerShell 2.0 or greater.
- A staging database that can run in a convenient Microsoft SQL Server instance. For example, this may be a database running on the inventory beacon, or on the XenApp server hosting the XenApp server agent, or on your central operations databases for FlexNet Manager Suite. For more about the requirements for this database,, see Creating the Staging Database.database

---

Tip: *As outlined in Architecture and Operation, it is possible to use additional command-line options for the XenApp server agent to remove the need for a staging database, and instead upload directly to an inventory beacon. Once the inventory beacon uploads the collected data to the pre-staging tables within the compliance database, the XenApp server adapter on the inventory beacon can continue operations as though these pre-staging tables were the staging database.*

- An inventory beacon (or multiple if required) that collects Active Directory data for the domain(s) where your XenApp server(s) are located.
- An inventory beacon (possibly the same as in the previous point) that can connect to your staging database and upload the inventory to the central FlexNet Manager Suite database. This process is performed by the XenApp server *adapter* on the inventory beacon.
- If you are using XenApp 6.x with the recommended EdgeSight server, an inventory beacon (almost invariably the same one as in the previous point) that can connect to the EdgeSight database.

Tip: *For XenApp 6.x with EdgeSight, also be sure to take inventory from your XenApp servers so that edition information is available to your license consumption calculations. This inventory may also be collected by an appropriate inventory beacon.*

- If you are using Citrix XenApp 7.6 (or later) or Citrix Virtual Apps 7.1808 (or later), an inventory beacon (usually the same one) that can connect to your XenDesktop database, and uploaded the imported data to your central operations databases.

Supported versions

6.0, 6.5, 7.5-7.9, 7.11–7.17 (XenApp), 7.1808–7.1909 (Citrix Virtual Apps)

2

Setting Up the XenApp Server Adapter

The XenApp server adapter is available for different versions of Citrix Virtual Apps (previously XenApp):

- For version 6.0 and 6.5, it augments information available from EdgeSight, particularly about streamed applications
- For version 7.5, it is the primary means of gathering inventory information from XenApp
- For version 7.6 and later, it again augments information about application usage collected from the XenDesktop database included with XenApp.

The adapter is automatically saved as part of the standard installation of FlexNet Manager Suite. Installation consists of five main activities, all described in the following topics:

- Setting up the staging database for the inventory that is collected (see Creating the Staging Database)
- Copying the appropriate folder for the adapter to your chosen XenApp server(s) (see Installing the XenApp Server Agent)
- Ensuring an appropriate account is available to run the adapter (also in Installing the XenApp Server Agent)
- Setting up a local scheduled task to run the agent as you require (see Create a Scheduled Task)
- Setting up a connection from an appropriate inventory beacon to the staging database (see Create Connections for Data Upload).

Creating the Staging Database

The staging database allows the XenApp server agent to drop collected data in a conveniently close location. From here, an inventory beacon collects the data for transfer to the central operations databases for FlexNet Manager Suite.

Tip: *If you are choosing to use direct upload by an inventory beacon instead of using a staging database, skip this topic.*

The staging database can be installed in any convenient Microsoft SQL Server 2012 (or later) database:

- If your inventory beacon is located on a SQL server, the staging database can be on the inventory beacon
- If your central operations databases for FlexNet Manager Suite are accessible from the XenApp server, the staging database tables are already present on your central SQL Server
- Where there is an inventory beacon with network access to your XenApp server (and Citrix Virtual Apps is running its database in SQL Server), the staging database can be installed into the same database as used by Citrix Virtual Apps
- If none of these suit, use any other SQL Server instance that allows network access both from the XenApp server agent on the XenApp server and from the inventory beacon.

It is a small footprint database (half a dozen tables and one stored procedure) that is size-limited by the scale of your Citrix Virtual Apps implementation, with data being replaced at each upload.

To create the staging database (using supplied script):

1. Use your browser to access the Product and License Center, access the current release, download the `Adapter Tools for` FlexNet Manager Suite `2019 R2.zip`, and save it to a convenient location (such as `C:\temp`) on a suitable server.
2. In your unzipped archive, navigate into the `\Citrix XenApp Server Agent` subdirectory.
3. Further navigate into the appropriate sub-folder for your version of Citrix Virtual Apps (noting that the **XenAppAgent** folder applies to XenApp and XenDesktop versions 7.5 and later, and to Citrix Virtual Apps and Virtual Desktops versions 7.1808 and later):
   - `XenAppAgent6`
   - `XenAppAgent65`
   - `XenAppAgent`

   Note: *If it happens that you have multiple different versions of Citrix Virtual Apps (for example, in different domains), you may use a single database for all versions of Citrix Virtual Apps. The following script is identical in all these folders.*
4. From your chosen folder, collect a copy of the database creation/update script `SetupXenAppAgentStagingDatabase.sql`.
5. On your selected SQL Server, drop a copy of this file, and execute it in SQL Server Administration Studio against your chosen database instance.

   An appropriate SQL Server database instance:
   - Is accessible from the XenApp server(s) running the XenApp server agent
   - Is accessible from the inventory beacon responsible for uploading collected inventory to FlexNet Manager Suite
   - Grants read/write access to the account running the scheduled task for the XenApp server adapter (if you choose to use Windows Authentication — if not, take note of the account name and password for database access that you will include in the database connection string)
   - Grants read access to the service account running the inventory beacon engine (if you choose to use Windows

Authentication — if not, take note of the account name and password for database access that you will include in the database connection string).

The script generates the SQL schema for the database, including creating the appropriate stored procedure. Take note of the connection string needed to connect to this database for your future reference.

Installing the XenApp Server Agent

This procedure assumes you have already located the installation folder where the XenApp server adapter awaits.

To install the XenApp server agent:

1. In your unzipped archive, navigate into the `\Citrix XenApp Server Agent` subdirectory.
2. Copy the appropriate sub-folder to match your installed version of Citrix Virtual Apps (formerly Citrix XenApp) (noting that the `XenAppAgent` folder applies to Citrix XenApp versions 7.5 or later and Citrix Virtual Apps versions 7.1808 or later):
   - `XenAppAgent6`
   - `XenAppAgent65`
   - `XenAppAgent`

   Tip: *If you have different versions of Citrix Virtual Apps (formerly XenApp) deployed in different domains, install the appropriate agents on the correct XenApp servers. Agents for different versions may connect to a single staging database(or upload collected data directly through a convenient inventory beacon).*
3. Using your network, a memory stick, or other available means, paste the entire folder into an appropriate location on your XenApp server(s).

   For example, you could paste the folder at the root level (such as `C:\XenAppAgent`). Keep in mind that for version 6 and 6.5, you must install the agent on a single XenApp server for each Citrix farm. For version 7.5 (or later), you may choose to install an agent on one XenApp server in each site, or to install on only one XenApp server that has network access (and credentials) for all your sites.
4. Ensure that, on your XenApp server (Delivery Controller), there is an account with sufficient privileges to run the agent in production.

   Such an account:
   - Can run a Windows scheduled task on the XenApp server where the agent is installed
   - Has read access to the file system on the XenApp server(s) where it is to collected inventory
   - Has read access to any file shares used to house XenApp packages (versions 6 and 6.5 only), App-V packages, or applications hosted in VDI images
   - For versions 6 and 6.5, is a Citrix Admin account (a limitation in the PowerShell API for those versions means that only a Citrix Admin can retrieve the list of XenApp servers in a farm)
   - For version 7.5 (or later), is a Citrix Read only admin account, with read access to the file systems of any other

XenApp servers sharing locally published applications for which inventory is to be collected

- Is recognized by the SQL Server hosting the shared database (if you prefer to use Windows Authentication for access to the staging database; otherwise, you may choose to include an account name and password in the connection string for the staging database).

Note: *If you are choosing direct transfer to an inventory beacon instead of using a staging database, you may either ensure that this same account has privileges to upload files to your inventory beacon, or use the separate parameters for username and password to identify an appropriate account for the transfer when running the XenApp server agent (for details, see XenApp Server Agent Command Line Options).*

Create a Scheduled Task

The XenApp server agent must be run locally on the XenApp server, where it collects inventory and transfers the data immediately either to the staging database, or to an inventory beacon for upload to the central compliance database. This is triggered by a Windows scheduled task on the XenApp server.

Because the XenApp server agent, as its first action for each inventory collection, clears all old data from the staging database, it is important that the XenApp server agent does not run at the same time as the inventory beacon collects data from the staging database (otherwise, corrupt or incomplete data may result). A buffer of 2 hours provides a good safety margin (depending on the scale of your Citrix Virtual Apps implementation).

Another consideration is that you want your Citrix Virtual Apps inventory uploaded to the central FlexNet Manager Suite database before the system import and compliance calculations take place. Typically, this process starts around 2am central server time. A two-hour upload buffer should be more than adequate.

These considerations suggest (within a single time zone) a collection schedule around 10pm, an inventory beacon connection around midnight, and everything in place for the nightly compliance calculation.

The process for setting up Windows scheduled tasks varies across different editions of Windows Server. The following example is for Windows Server 2012. Adjust for your XenApp server's conditions.

To create the scheduled task (Windows Server 2012 example):

1. In Windows Explorer, navigate to **Control Panel** > **System and Security** > **Administrative Tools**, and double-click **Task Scheduler**.

   The **Task Scheduler** window appears.

2. In the navigation tree on the left, select **Task Scheduler Library**, and then in the **Actions** list on the right, click **New Folder...**.

   A dialog appears for entering the folder name.

A suggested value is FlexNet Manager Suite.

3. Click **OK**, and select the new folder in the navigation tree.
4. Select **Action** > **Create Task...**.

   The **Create Task** dialog appears.
5. Enter an appropriate **Name**, such as XenApp server agent, and add any **Description** to help future maintenance of this task.

   Your description may be something like `Collects application and access information from a Citrix XenApp server and transfers this information to a staging database`.
6. Click **Change User or Group...**.

   The **Select User, Service Account, or Group** dialog appears.

7. Enter the account name that is to run the scheduled task, and click **OK**.

   This is the account you identified in Installing the XenApp Server Agent, and you can check the requirements for this account there.

8. Further down in the **Security options** group, select **Run whether user is logged in or not**.

9. Switch to the **Triggers** tab, and click **New...**.

   The **New Trigger** dialog appears.

10. Ensure that the default setting **Begin the task** `On a schedule` is selected, set the parameters for the schedule, and from the **Advanced settings** group, be sure that **Enabled** is selected.

    The suggested schedule is daily at 10pm local time, but be sure that this suits the upload procedures for your enterprise.

11. Switch to the **Action** tab, and click **New...**.

    The **New Action** dialog appears.

12. Ensure that the default **Action**, `Start a program`, is selected, and browse to your local copy of `FnmpXenAppServerAgent.exe`.

13. In the **Add arguments (optional)** field, specify all the command-line arguments you need for the agent.

    All command line arguments are documented in XenApp Server Agent Command Line Options. For common implementations, you will either define the connection string to your staging database, or provide the URL, account, and password for direct transfer to an inventory beacon. In addition, if you are using Citrix XenApp 7.5 (or later), or Citrix Virtual Apps 7.1808 (or later) and want a single agent to collect inventory from multiple

servers, you need the option to define those servers.

Example 1: Command line arguments to connect to your staging database.

```
-d "Data Source=192.168.13.38;Initial Catalog=MyStaging;
            User ID=accountName;Password=password"
```

Example 2: For Citrix XenApp 7.5 (or later) and Citrix Virtual Apps 7.1808 (or later), accessing multiple XenApp servers and recording their details in a common staging database (all on one line):

```
-d "Data Source=192.168.13.38;Initial Catalog=MyStaging;
            User ID=accountName;Password=password"
 -s "localhost, xda01.fqdn.com"
```

14. Click **OK**.
15. Optionally, make any preferred adjustments to the **Conditions** or **Settings** tabs (normally the defaults are acceptable).
16. Click **OK** to close the **Create Task** dialog.

    The new task appears in the list of scheduled tasks for this server.
17. Right-click the new task, and click **Run** in the context menu.

    This checks that the scheduled task completes successfully.
18. Validate operations in the following ways:
    - Review the log file (`FnmpXenAppAgent.log` in the same folder as the agent's executable file) for any errors or warning messages.
    - Use Microsoft SQL Server Management Studio to check the contents of the staging database.

Create Connections for Data Upload

The XenApp server agent gathers inventory information from your XenApp server, and does either of two things with the collected data:

- It saves it in a separate staging database. In this case, the XenApp server *adapter* (running on the inventory beacon) takes over, and is responsible for collecting the staged data, de-duplicating and normalizing the data, and uploading it to a set of `Imported*` tables in the central compliance database. The adapter requires a connection to the staging database, and a schedule to trigger collection of inventory data from the staging database.
- The XenApp server agent transfers collected data directly to an inventory beacon. In this case, the inventory beacon uploads the data to pre-staging tables in the central compliance database. Thereafter, the XenApp server *adapter* must be triggered to de-duplicate and normalize the data, loading results into the `Imported*` staging tables in the same compliance database. The adapter requires a connection to the compliance database, and a schedule to trigger processing of inventory data from the pre-staging tables to the `Imported*` staging tables.

In addition, with the exception of XenApp version 7.5, a connection to the appropriate database is strongly recommended, as this allows tracking who is actually using applications:

- If you are using XenApp 6.0 or 6.5, the connection is to your EdgeSight server database
- If you are using XenApp 7.6 (or later) or Citrix Virtual Apps 7.1808 (or later), the connection is to the XenDesktop database included with Citrix Virtual Apps (formerly Citrix XenApp).

Perform this process on the inventory beacon.

To create connections for data upload:

1. Log in to your selected inventory beacon.

   Tip: *Starting the FlexNet Beacon interface requires that you are logged in with administrator privileges.*

2. If you have not already specified a schedule (or two) that can be linked to the connection(s) you are about to create, it's convenient to do so now.

   There are two connections for each version of Citrix Virtual Apps (formerly XenApp) except version 7.5, which needs only one. Remember that you already decided on the schedule for data collection on the XenApp server (see Create a Scheduled Task), and the schedules on the inventory beacon need to tie in with that plan. For example, you might schedule the connection to the staging database at midnight, and the secondary connection for usage data at 12:15am. For details about creating a schedule on the inventory beacon, see *FlexNet Manager Suite Help > Inventory Beacons > Scheduling Page > Creating a Data Gathering Schedule*.

3. In the navigation pane on the left, select the **Inventory systems** page, and towards the bottom of the page, click **New...**.

   The **Create SQL Source Connection** dialog opens.

   Tip: *The **New...** button defaults to creating a connection for Microsoft SQL Server. If you use the down arrow on the split button, you can choose between `SQL Server`, `Spreadsheet`, `PowerShell`, and `Other` connections. However, while you are creating a connection to a Microsoft SQL Server database (regardless of the **Source Type** of the connection), use only the `SQL Server` option.*

4. Complete the values in the dialog, as follows:

Control	Comments
Connection Name	A descriptive name for this connection, such as `Citrix Virtual Apps` *`ServerName`* `Staging DB` (or, if you are using the inventory beacon upload process, perhaps something like `Citrix Virtual Apps Data Normalization`).
Source Type	Select **Citrix XenApp (Server Agent)**. (Don't be confused by **Citrix XenApp (EdgeSight)**, which you may use shortly.)

Control	Comments
Server	Type the server name or IP address. Use the special value `(localhost)` if the database is installed on this same inventory beacon server. If the database instance you need is not the default one on the server you identify, add the instance name, separated with a backslash character. Example: `(localhost)\myInstance` ***Note:*** *If you are using the inventory beacon upload process, so that the collected data has already been uploaded to the pre-staging tables in the central compliance database, this connection is to your central database server running that compliance database.*
Authentication	Select one of: • **Windows Authentication** — Select this option to use standard Windows authentication to access the database server. The credentials of the account (on the inventory beacon) running the scheduled task for importing inventory are used to access the SQL Server database. This account must be added to an Active Directory security group that has access to the database. • **Windows (specific account)** — Use the following two fields (enabled when you make this choice) to specify an account on the inventory beacon that can make a connection to the SQL database. • **SQL Authentication** — Use the following two fields to specify an account and password registered as a user with database access on SQL Server . This account is used to access the database, regardless of the local account running the scheduled task on the inventory beacon server. ***Tip:*** *The account used needs read-only privileges.*
Username	The account name used for **SQL authentication**, or **Windows (specific account)**. (Not required for **Windows Authentication**.)
Password	The password for the account name required for **SQL authentication**, or **Windows (specific account)**. (Not required for **Windows Authentication**.)
Database	Enter the name of the database, or use the pull-down list to select from database names automatically detected on your specified server.

Control	Comments
Connection is in test mode (do not import results)	For inventory beacons that are free-standing (and not co-located with your application server, or in larger implementations your inventory server), this controls the upload of data collected from this connection and its import to FlexNet Manager Suite: • When this check box is clear, the connection is in production mode, and data collected through this adapter is uploaded to the central server and (in due course) imported into the database there. • When the check box is set: ◦ The adapter for this connection is exercised, with data written to the intermediate file in the staging folder on the inventory beacon (*%CommonAppData%*`\Flexera Software\Beacon\IntermediateData`) ◦ The immediate upload that normally follows data collection is suppressed, so that you can inspect the contents of the file ◦ The catch-up process that retries stalled uploads, normally scheduled overnight, runs as usual and uploads the file to the central server ◦ At the central server, the file contents are discarded (and not imported into the central database). ***Tip:*** *When an inventory beacon is co-located with your application server, data from the connection is read directly into staging tables in the database, and no intermediate file is saved.*
Overlapping Inventory Filter	This control does not apply to Citrix Virtual Apps inventory records, and you may leave it at the default setting.

5. Click **Test Connection**.
   - If the inventory beacon can successfully connect to the nominated database using the details supplied, a `Database connection succeeded` message displays. Click **OK** to close the message. Click **Save** to complete the addition. The connection is added to (or updated in) the list.
   - If the inventory beacon cannot connect, a `Database connection failed` message is displayed, with information about why that connection could not be made. Click **OK** to close the message. Edit the connection details and retest the connection.

   You cannot save the connection details if the connection test fails. If you cannot get the connection test to succeed, click **Cancel** to cancel the addition of these connection details.
6. When the first connection to the database is successful, and if you are using any Citrix Virtual Apps (XenApp) version other than 7.5, repeat the steps above to define a second connection to the usage database (EdgeSight database for version 6.x, and Citrix Virtual Desktops (XenDesktop) database for version 7.5 and later). This time through, for all these versions set the **Source Type** control to **Citrix XenApp (EdgeSight)** (including for Citrix Virtual Apps (XenApp) version 7.5 and later).

7. On the **Inventory systems** page, from the list of connections select the one to the staging database you created first in this process, and below the list, click **Schedule...**.
8. In the resulting dialog, choose the schedule you wish to use for this connection, and then click **OK** to close the dialog, and **Save** to apply the selected schedule to your connection.
9. For any Citrix Virtual Apps (XenApp) version other than 7.5, on the **Inventory systems** page, from the list of connections select the EdgeSight connection, and repeat the scheduling process, choosing the second schedule you created.

Don't forget, as you move this XenApp server adapter into production, to ensure that **Connection is in test mode (do not import results)** is clear (not checked).

3

Command-Line Options

For those times when you want to control execution of the command-line agent directly, this chapter covers all options.

XenApp Server Agent Command Line Options

XenApp server agent (`FnmpXenAppAgent.exe`) is a command line tool which runs regularly on a Citrix XenApp server (for Citrix Virtual Apps) to determine which end-users have the right to run applications through that server. The data it collects is sent back to FlexNet Manager Suite and processed further once the inventory import process runs. Here are the details for manual operation from the command line, or configuring a scheduled task.

Syntax:
`FnmpXenAppAgent.exe` [*options*...]

Options

`-d` *`connection`*
`-f` *`site_name_override`*
`-h`
`-i` *`true/false`*
`-pass` *`password`*
`-o` *`output_file`*
`-r` *`true/false`*
`-s` *`servers`*
`-t` *`timeout`*
`-url` *`beacon_address`*
`-user` *`username`*
`-v` *`0|1`*
where

`-d` *`connection`*	A database connection string to your staging database. Refer to http://www.connectionstrings.com/sql-server-2008 for some examples. If you do not have a staging database but connect directly to your central compliance database, create the connection string for that database. ***Note:*** *Do not use more than one of the `-d` option, the `-o` option, and the `-url` option at the same time.*
`-f` *`site_name_override`*	Force an override of the default Citrix Virtual Apps farm/site name.
`-h`	Displays usage for the XenApp server agent.
`-i` *`true/false`*	(Default false.) Ignore errors. Used only for debugging purposes so that the adapter runs end-to-end and logs all issues in the log file (in the same directory as the agent executable).
`-pass` *`password`*	The password for the account used for data upload to an inventory beacon. ***Note:*** *This option is used in conjunction with the `-user` and `-url` options. If you do not specify the `-pass` option, the credentials for the account running the XenApp server agent are used.*
`-o` *`output_file`*	The full path of a file to store the output of the XenApp server agent as it runs. Use such an output file for debugging purposes, to see the content collected by the agent. The output file is not required for normal operations (the collected data is uploaded directly to the staging database). ***Note:*** *Do not use more than one of the `-d` option, the `-o` option, and the `-url` option at the same time.*

`-r` *`true/false`*	Option only for Citrix XenApp 7.5 (or later) or Citrix Virtual Apps 7.1808 (or later). Values may be: • `false` - (default) Disable remote registry environment variable resolution, and enable *local* variable resolution on the XenApp server. • `true` - Direct the XenApp server agent to remotely read the Windows registry on the Citrix Virtual Apps *client* machine, using values there to resolve environment variables embedded in file paths. As background, when a file path in Citrix contains a Windows environment variable (for example, `%ProgramFiles%`), the variables can have different values on the server and on the client device. This switch lets you choose the value that is resolved locally on the XenApp server, or choose the environment variables remotely read from the registry of the Citrix Virtual Apps client device. Resolving environment variables using this command requires both of the following: • The **Remote Registry** service must be started (status of `Running`) on the client machine in order to resolve remote environment variables • The account executing the XenApp server agent must exist on the Citrix Virtual Apps client machine to properly resolve environment variables using this command.
`-s` *`servers`*	Option only for Citrix XenApp 7.5 (or later) or Citrix Virtual Apps 7.1808 (or later). For both 6.5 (which does not support this option) and 7.5 or later (where the option may be omitted), the XenApp server agent assumes that it is running on the XenApp server from which it is to collect inventory. In both systems, therefore, you may handle multiple XenApp servers by installing the XenApp server agent on each one. When the agent is installed locally on the XenApp server from which it gathers inventory, omit this option. In Citrix XenApp 7.5 (or later) or Citrix Virtual Apps 7.1808 (or later), you have the option for a XenApp server agent installed on a single XenApp server to collect the inventory for all XenApp servers in a server farm. To do this, create a comma-separated list of fully-qualified domain names or IP addresses for all the servers from which this agent should collect inventory. Inside such a list, use the keyword `localhost` to include the server on which the XenApp server agent is installed. ***Note:*** *The account executing the XenApp server agent must have permissions to connect to each of the servers named in your list.*
`-t` *`timeout`*	The timeout period (in seconds) when connecting to the staging database (default 600 seconds). If the timeout expires, the upload fails, and the data collected from the XenApp servers on this occasion is lost. An entry is made in the log file (in the same directory as the agent executable) to record the failed connection.

`-url` *`beacon_address`*	The URL of the inventory beacon where the XenApp server agent uploads collected data. ***Note:*** *Do not use more than one of the `-d` option, the `-o` option, and the `-url` option at the same time.* ***Note:*** *This option is used in conjunction with the `-pass` and `-user` options. If you do not specify the `-pass` and `-user` options, the credentials for the account running the XenApp server agent are used.*
`-user` *`username`*	The username on the inventory beacon, the account which allows data upload to the inventory beacon. ***Note:*** *This option is used in conjunction with the `-pass` and `-url` options. If you do not specify the `-user` option, the credentials for the account running the XenApp server agent are used.*
`-v` *`0|1`*	(Default 1). Sets logging messages to verbose mode. For less information, specify `-v 0`.

Examples

The following examples are split across several lines for readability. Run each example on a single command line.

The examples depend on which form of authentication you have chosen to use:

- Windows Authentication, using a Windows (Active Directory) account that is recognized by the SQL Server hosting your staging database
- A separate SQL Server account specified in the database connection string.

(You decided on your approach in Installing the XenApp Server Agent.) These two approaches are mutually exclusive, so each example applies to one or the other, but not both.

Use Windows Authentication (for the account running the scheduled task, or the command line) to connect to the staging database with a ten minute (600 second) timeout:

```
FnmpXenAppAgent.exe
        -d "Server=192.168.13.38;Database=MyStaging;Trusted_Connection=yes"
        -t=600
```

Using SQL Server authentication, identify a particular user name and password to connect to the staging database:

```
FnmpXenAppAgent.exe
        -d "Data Source=192.168.13.38;Initial Catalog=MyStaging;
            User ID=accountName;Password=password"
```

Using Windows Authentication (with the relevant account known to both SQL Server hosts) to collect inventory in XenApp 7.5 (or later) or Citrix Virtual Apps 7.1808 (or later) from two servers, saving the collected data in an XML file for debugging purposes:

```
FnmpXenAppAgent.exe
        -s "localhost, xda01.fqdn.com"
        -o "c:\XenAppTest.xml"
```

Have the XenApp server agent upload collected data to an inventory beacon:

```
FnmpXenAppAgent.exe
        -url http://beacon01.example.com -user beaconAccountName -pass password
```

4

Validation, Troubleshooting, and Limitations

This chapter may assist in diagnosing operations of the adapter, as well as explaining certain inherent limitations in its operation.

Validation and Problem Solving

Because the XenApp server adapter has a number of moving parts, validation and problem solving may also involve multiple steps.

After initial implementation, the simplest validation is simply to inspect the additional installer evidence (for App-V applications) and file evidence (for VDI applications) that are collected by the adapter, and displayed in the web interface for FlexNet Manager Suite. Keep in mind that to complete the end-to-end process, you may need to

- Manually link the evidence to an appropriate application
- Ensure that the application is linked to a suitable license
- Record entitlements on the license, typically by linking purchases to it.

Also remember that you must be importing information from Active Directory prior to importing inventory through the XenApp server adapter.

When more detailed analysis is required, you may use the following checks.

XenApp server agent

The XenApp server agent, installed on your XenApp server, records a log file in the same folder where it is installed. The log file is replaced at each inventory collection (that is, each time the scheduled task triggers the agent). Review the log for details of any problems. To increase the level of detail, run the agent with the command-line option `-v 1`.

To see all the information that the XenApp server agent has collected, run the agent without a `-d` option (the path to the staging database) and instead using a `-o` option with a path to a convenient local folder. This saves a plain-text XML file of the collected inventory that you can inspect in your preferred text editor. This is a valuable check point when some inventory is being returned, but particular expected applications seem to be missing. If these are missing from a

file output with the `-o` option, look for reasons preventing agent access to the source information. Examples might include credentials or access rights to folders containing packages.

If you use the `-o` option, don't forget to replace it with the `-d` option for normal operations!

Staging database

When records missing from the web interface for FlexNet Manager Suite are present in the output of the XenApp server agent (see previous section), next use Microsoft SQL Server Administration Studio to inspect the contents of the staging database. Recall that the contents of the staging database are over-written with each inventory collection by the XenApp server agent. This means that there may be legitimate differences between an output file obtained in the previous section, and the database contents examined in this section. Such differences may come about if there is additional access granted to Citrix Virtual Apps (formerly XenApp) applications (the source data) in between the time the agent is run to output the test file, and the time it is run to populate the staging database. In general, however, there should be a high degree of correlation between the two data sets.

The inventory beacon intermediate file

Next, you can validate the data set that the inventory beacon collects from the staging database. Simply trigger the connection to the staging database in test mode, as described in Create Connections for Data Upload. This allows you to inspect the zip archive, which would otherwise be uploaded to the central server, in `%CommonAppData%\Flexera Software\Beacon\IntermediateData` on the inventory beacon. Provided that you make comparisons before the next run of the XenApp server agent, you should find 1:1 correspondence between the data in the staging database and the intermediate file.

Uploads and imports

If the required data has made it into the intermediate file on the inventory beacon, you may need to debug uploads from the inventory beacon to the central server (for example, see *FlexNet Manager Suite Help > Inventory Beacons > Inventory Beacon Reference > Troubleshooting: Inventory Not Uploading*).

Keep in mind that there is a delay between the upload from the inventory beacon to the central server, and the appearance of the data in the web interface for FlexNet Manager Suite. There must be an inventory import and compliance calculation that occurs between these two events. If you are a member of the Administrator role, in the web interface you may manually trigger an import and compliance calculation.

Limitations

The following limitations apply to the XenApp server adapter:

- Information about who has access to applications on the one hand, and about who actually uses applications on the other, is collected separately. Usage data relies on a Citrix EdgeSight server (for XenApp 6.x) or the Citrix Virtual Desktops (XenDesktop) database included with XenApp version 7.5 (or later) and Citrix Virtual Apps version 7.1808 (or later). Since XenApp 7.5 does not support it, no usage information is available for this version; and if you are using any other version without the appropriate database connection, again no usage information is available.
- When the XenApp server agent connects to a VDI image to read the applications it contains, and the image is not currently running, the agent attempts to start up a server running the image for interrogation (and will shut it down again afterward). This relies on the remote support of power actions (on and off, and so on) for the master image.

Where these are not available, or the Citrix Virtual Desktops (formerly XenDesktop) do not have sufficient resources to spin up another image, the start-up attempt fails. In these cases, the agent imports the name of the executable, but it is missing the version, company, and description details. These cases appear in the list of file evidence with the executable name, but with the name, version, company, and description columns blank.

- Citrix Virtual Apps (formerly Citrix XenApp) supports "application delivery from a remote PC" and "application delivery from the cloud". Neither of these is supported by the XenApp server adapter, and no inventory is returned for such cases.
- To improve performance compared with earlier versions of the XenApp server adapter, the XenApp server agent no longer collects user names and computer names through the XenApp server. Instead, it collects only the Active Directory security IDs (SIDs) from Citrix Virtual Apps (formerly XenApp), and relies on the separate import from Active Directory to flesh out the SIDs with more complete identities. This has two immediate implications:
  - If you are upgrading from an earlier version of the XenApp server adapter, the data from the access control lists (ACLs) on the XenApp server is removed on upgrade. An import from Active Directory is then required to populate the SIDs and other identity details. After the first inventory import in the upgraded system, the access data is repopulated.
  - If Active Directory information is not imported by an inventory beacon for the domain(s) in which the XenApp servers (the ones hosting the XenApp server agent) are located, users and computers newly registered in Active Directory (since the last import of Active Directory data) will not be recognized or displayed in FlexNet Manager Suite.
- For XenApp version 7.6 (and later) and Citrix Virtual Apps version 7.1808 (and later), application usage information is available only for App-V packages and applications, and streaming profile applications. Usage information is not available for file-based applications, including VDI applications delivered through Citrix Virtual Apps.

5

Database Impacts

This chapter provides an overview of database tables within FlexNet Manager Suite that are affected by the adapter. These brief notes can be augmented by reviewing the FlexNet Manager Suite Schema Reference PDF file for the current release.

Affected Database Tables

The XenApp server adapter causes data to be imported to the following database tables in the operations databases for FlexNet Manager Suite (specifically the compliance database). You may prepare custom reports against these:

- `ComplianceDomain` records the domain of the XenApp server where the XenApp server agent is installed. If the record does not already exist, on import both the FQDN and the flat name are populated.
- `ComplianceUser` records are created for any user names identified in the Citrix Virtual Apps inventory but not previously in the operations databases. In these new records, only the `UserName`, `SAMAccountName`, and domain are available and saved (with `ComplianceUserID` calculated automatically). (The domain is a link to the `ComplianceDomain` table, which once again is updated as required with any new records. Such updates should be rare, since domains should be identified from Active Directory.) Because the `ComplianceUser` records created from Citrix Virtual Apps inventory imports are far from complete, you may want to enhance these with additional information.
- `ComplianceComputer` records may be created if a `ComplianceUser` record is created (or identified) that has no link to an existing `ComplianceComputer` record. For many types of license, the consumption record is linked to an individual `ComplianceComputer`, so when it is impossible to identify a computer for a particular `ComplianceUser`, a new skeleton computer record is created. These have a computer name of the form

  ```
  UserName (Remote)
  ```

  and have their type shown as `Remote Device`. (This type is identified through a foreign key to the `ComplianceComputerType` table.) They are also linked to the `ComplianceUser` for whom they are created.

  Tip: *If, in future, inventory from another source identifies the same `ComplianceUser` as either the assigned user or the calculated user for a computer identified by that inventory, then this placeholder record for the remote device is removed, and any license consumption recorded against it is moved to the newly-identified (real)*

computer that belongs with the user.

- For App-V applications managed by Citrix XenApp 7.5 (or later) or Citrix Virtual Apps 7.1808 (or later), or any Citrix Virtual Apps applications managed by version 6.x:
  - `InstallerEvidence` records are created as the primary inventory data for the use of those applications.
  - `InstalledInstallerEvidence` records are create to link that evidence to the appropriate `ComplianceComputer` records.
  - Where the application name, version, and publisher in the `InstallerEvidence` table match an existing evidence rule for an application, an entry is created in the `SoftwareTitleInstallerEvidence` table to link (by foreign keys) the installer evidence to the application record in the `SoftwareTitle` table.
- For VDI applications managed by Citrix XenApp 7.5 (or later) or Citrix Virtual Apps 7.1808 (or later):
  - `FileEvidence` records are created as the primary inventory data for the use of those applications.
  - `InstalledFileEvidence` records are create to link that evidence to the appropriate `ComplianceComputer` records.
  - Where the file name, version, company, and description in the `FileEvidence` table match an existing evidence rule for an application, an entry is created in the `SoftwareTitleFileEvidence` table to link (by foreign keys) the file evidence to the application record in the `SoftwareTitle` table.
- `InstalledApplications` records are created for any applications identified in the `SoftwareTitle` table, linking the application to the `ComplianceComputer` record.

XV

The Inventory Adapter Studio

Inventory Adapter Studio is a tool to develop adapters for custom inventory gathering into FlexNet Manager Suite.

This section covers the use of Inventory Adapter Studio and the management of the adapters you develop.

1

What Is Inventory Adapter Studio?

As well as gathering inventory directly from devices in your computing estate, FlexNet Manager Suite can also import inventory gathered by other software and hardware inventory tools. FlexNet Manager Suite ships with several factory-supplied adapters that read data from inventory tools such as Microsoft SCCM or IBM's ILMT. The Inventory Adapter Studio enables modification of these existing adapters; but more importantly, it allows creation of new adapters to connect with systems not supported out of the box.

Note: *The Inventory Adapter Studio is tailored specifically to building adapters for inventory tools. FlexNet Manager Suite is also able to import additional business-related data that influences license compliance calculations, but these connectors are built with the separate Business Adapter Studio.*

The Inventory Adapter Studio provides the following benefits:

- A graphical user interface that simplifies creation and editing of the underlying XML files that define the adapters.
- Template adapters, which include approved code for data transformation and import into the FlexNet Manager Suite operations database. This allows you to focus exclusively on the data gathering aspects of the adapter.
- Syntax highlighting, and highlighting of steps that require further editing.
- Data isolation by hiding test connections from your production implementation of FlexNet Manager Suite.
- Reduced testing effort, with a built-in filter to limit the number of records processed in a testing cycle.
- Context sensitive error reporting, with progress monitoring of each statement in the user interface.
- Detailed error reporting and tracing.

At the end of the section on the Inventory Adapter Studio, the object model for inventory adapters running on your inventory beacon in disconnected mode is fully documented.

2

Cautions, Prerequisites, and References

Caution: *Be aware that the Inventory Adapter Studio is an advanced tool, and incorrect use can result in data changes in your FlexNet Manager Suite environment that will affect your license position. If you are uncomfortable with performing these changes yourself, please contact the Flexera services team.*

The Inventory Adapter Studio has the following requirements:

- **Location.** The Inventory Adapter Studio may be installed either:
  - On an inventory beacon that will run the downstream functions of the adapter, connecting to the third-party inventory source within your enterprise. The intermediate files collected on this inventory beacon are then uploaded automatically to your central batch server. (This is called "disconnected mode" as it does not allow direct access to the underlying database.)
  - On the same central batch server as will perform the import process. In smaller implementations, use the server fulfilling that role. (This is called "connected mode", since it allows the inventory connector run from the central server to have direct access to your central database.)
- **File access.** On the server orbeacon where it is installed, Inventory Adapter Studio requires permission to create and edit files in the `C:\ProgramData\Flexera Software\Compliance\ImportProcedures` directory.
- **Downstream database access.** Inventory Adapter Studio requires permission to access the source databases. Read-only access may be sufficient, depending on how you write the adapter: many adapters write temporary tables on the source database to allow joins with existing data (for example, to create differential inventory of changes since the last import). Clearly, testing adapters such as these means that Inventory Adapter Studio must have full access to the source database.
- **Upstream database access.** When installed on the batch server, Inventory Adapter Studio needs both read and write access to the operations database of FlexNet Manager Suite. For large-scale systems where the operations database is split out to separate servers for the inventory database and compliance database, Inventory Adapter Studio requires read/write access to both databases. In contrast, when Inventory Adapter Studio (or any completed adapter) runs from an inventory beacon, the intermediate files are uploaded automatically to the central server, and automatically imported into the database.

Operator Requirements

To use the Inventory Adapter Studio successfully, you will need:

- A working knowledge of SQL, so that you can modify queries in the templates provided.
- Some data analysis experience.
- To edit the SQL queries to collect data from the source database(s), you need to know where to get the data in those source databases is located. This probably requires read access to the database and also access to the inventory tool's user interface for data validation.
- Direct access to the FlexNet Manager Suite server. This may be on the server console directly, or using terminal services.
- A detailed working knowledge of the FlexNet Manager Suite product. This is required so that data imports can be verified, along with the expected application installations.
- Good insight into database schemas.

Restrictions

Several inventory adapters are supplied as standard with FlexNet Manager Suite (these are sometimes referred to as "Tier 1" adapters). These are installed on your central server, and are automatically downloaded to inventory beacons as required. This means:

- You cannot modify Tier 1 inventory adapters on an inventory beacon.
- You cannot store anything else in the same folder on the inventory beacon used to store Tier 1 adapters distributed by the central server. Changes are always removed automatically within a short time, keeping the distributed adapters safe and in line with the latest versions stored on the central server.
- You can, however, modify inventory adapters (using the Inventory Adapter Studio) on your central application server. These then become your latest version and are automatically distributed to your inventory beacons.
- You can install your custom inventory adapters into `%CommonAppData%\Flexera Software\Compliance\ImportProcedures\CustomInventory` on your batch server. As with Tier 1 adapters, adapters in this folder are automatically distributed to (and can be exercised on) all your inventory beacons.
- You can also create custom inventory adapters on inventory beacons, using the Inventory Adapter Studio and the object adapter model described in the following topics. These are stored separately, and are not over-written by downloads from the central server. This also means that a custom inventory adapter is by nature restricted to the inventory beacon on which it is created. However, if you need the identical adapter operating from multiple inventory beacons, you can manually copy the adapter between beacons, always using the same file path (`%CommonAppData%\Flexera Software\Compliance\ImportProcedures\ObjectAdapters`).

3

The Inventory Adapter Studio Interface

The Inventory Adapter Studio has the following key areas in its interface:

Element	Purpose
Toolbar	• Creates new adapters or open existing ones. • Manages database connections. • Saves changes. • Specifies the database connection to work on. • Specifies the data size limits to apply when testing.

Element	Purpose
Step Explorer	Shows the steps in the currently open adapter. These open the edit panels on the right. • Import execution status will show as icons in this element. • Bold steps in the templates show where user customization is required; other steps have been completed by Flexera. • Steps may be added, deleted or have their execution order changed using the toolbar in the step explorer.
Step editor	Shows: • The name of the step and its settings. • The script in the step, with a Run button for testing. • The logs, showing import results. • The Result panel, which shows datasets from your SQL queries.
Status bar	Shows import progress.

Each section is discussed in more detail in the following topics.

Toolbar

The toolbar contains the following controls:

New button

A button that launches the **Create New Adapter** dialog.

Open button

A button that launches the **Open Existing Adapter** dialog. This allows browsing of all custom and factory-supplied adapters.

Save button

The Save button saves all files in the adapter that is being edited. This includes changes due to steps being moved in the Step Explorer, or versions being changed in the Version field on the toolbar.

Source Connection control

The drop-down portion of this control allows selection of an existing database connection. New connections can be created and existing connections may be edited using the '…' button, which launches the **Select Source Connection**

dialog.

Version control

This control shows the version of the currently selected source connection. This version is evaluated by executing a query against the source database. Clicking the drop-down button displays a dialog that allows you to change this query for an adapter. Clicking the **Retrieve Source Version** button will execute the query and show the results in the dialog.

Use of this control is particularly important if your adapter supports importing inventory data from multiple versions of the same system. This usually occurs as enterprises upgrade systems over time.

Limit Number of Computers control

This control is checked and enabled by default for test connections. When set to a value, it limits the number of computers read by an adapter.

Full Import button

This button launches the **Full Import** dialog, which allows you to execute your adapter end to end and check your results in FlexNet Manager Suite.

Options

There are several options that apply to the Run buttons on the Edit panel. These control the way the Compliance Importer executes your adapter and correspond to command line arguments.

Options include:

- **Run optional readers** — the next run command will exercise steps marked with the **Step is optional** attribute.
- **Force full import** — the next run command includes steps marked with the **Runs on full import** attribute.

Tip: *When the attribute values prevent the execution of the step in the next run, it is grayed out with a strike-through in the step explorer.*

Step Explorer

The step explorer shows the procedures in the Compliance Importer, and the steps within those procedures that will be executed for the current adapter.

The step explorer is a docking control and may be moved within the Adapter Studio interface. It also has a column that shows the file a step is being saved to.

Expanding one of the top level procedures shows all the steps within it.

Bold steps and procedures are parts of the template requiring your customization. There are queries in that part of the template that need to be replaced with code that applies to your data source.

Figure 6: Collapsed (left) and expanded, with bold steps requiring customization. Custom SQL and data transfer steps visible.

The toolbar allows steps to be added, removed, edited and to change their execution order. There are two types of steps that may be added:

- Custom SQL steps have a yellow database icon and run commands on the source or target database.

- Data transfer steps have a blue icon with white arrows, and copy data from one database to another using a bulk transfer.

When testing an adapter, the step explorer also shows the status of the current run with green and red icons.

When the attribute values of a step will prevent it being executed for the version of the current connection, it will be grayed out with a strike-through in the step explorer.

Edit panel

The edit panel consists of three main areas:

- A properties section
- An SQL script
- A log message and result panel.

Properties section

Step name

This is the name of the step, and is shown on the step explorer as well as the top of the edit panel tab. It is best to choose a descriptive name to simplify future maintenance.

Execution

Possible values for custom steps are:

Value	Supported modes	Step type	Notes
`ExecuteOnSource`	*Connected*, *disconnected*, *split*	Custom SQL	The SQL script will run on the source database.
`ExecuteOnTarget`	Connected, split	Custom SQL	The SQL script will run on the FlexNet Manager Suite database.
`GetVariableFromSource`	Connected, disconnected, split	Custom SQL	Allows you to declare, and get a value for, a custom SQL variable populated from the source database. This variable and its value are automatically in scope and available for all subsequent steps in this inventory adapter, alongside the standard variable `ComplianceConnectionID`.

Value	Supported modes	Step type	Notes
`SourceToObject`	Disconnected	Data Transfer	The SQL script reads from the source database, and writes approved data directly to an intermediate package saved on the inventory beacon. A separate process uploads this intermediate package to the application server, where another scheduled process imports the data into the operations database. `SourceToObject` steps may only write to the predefined objects displayed in the Inventory Adapter Studio.
`SourceToTarget`	Connected, split	Data Transfer	The SQL script runs on the source database, transferring resulting data to the central operations database. The destination table in the operations database must be identified in the next field.
`TargetToSource`	Connected, split	Data Transfer	The SQL script will run on the central operations database, transferring resulting data to the source database. The destination table in the source database must be identified in the next field. A typical use is to populate a temporary table on the source database with the current state of inventory in the central store. This allows decentralized processing (on the source database server) to produce differential inventory for upload.

Destination table

This field only applies to data transfer steps in adapters running in connected or split modes. It specifies the database table into which the results of the SQL query will be bulk copied. The table data types must exactly match the columns of the query, and columns must be in the correct order. The provided templates contain data conversions and string trimming as well as the correct ordering to make this task as easy as possible.

File

This is the file name where the step is saved. In the templates it is specified for you, and has little impact on the execution of the adapter.

Step is optional

An optional step will not be executed by default when the Compliance Importer is run. The only example of this in the factory-supplied adapters is when file information that does not match the Application Recognition Library is returned. Use this when the data returned by the step is not needed for critical tasks.

Runs on full import

By default, adapters perform differential imports and only update computer records that have changed since the last import. The `#RelevantComputers` temporary table in the templates implements this feature. This flag is set for steps that are designed to override this functionality. The provided templates have one step with this option set, and causes all computers to be updated instead of the differential import.

Version from

This is the first version field, and it causes the step to only execute when the **Version** field in the toolbar equals the specified value or higher. The version format must be in the 1.2.3.4 form.

Up to version/Before version

This is the second version field, and it causes the step to only execute when the **Version** field in the toolbar is less than the specified value (less than or equal in the case of **Up to Version**). The version format must be in the 1.2.3.4 form.

SQL Script section

Run current script

This button executes the script for the current step. The SQL is executed and any result sets is displayed in the **Result** tab. You may execute multiple queries and return multiple result sets. You may execute parts of the script by selecting them before using the button.

Different step types execute on the following databases by default: for details, see the **Execution** field in the *Properties section* above.

There is a right-click menu available in the script edit panel that allows you to specify execution of the script on a different database.

Run from beginning

This button executes the adapter from the beginning up to the current step. Once the current step is reached, execution is terminated, but the database connections are left open so you can run queries to inspect database values as desired. The results will be in the **Log Message** tab.

Stop

This button stops an adapter that is in the process of running.

SQL Script

This area contains the SQL scripts that make up the adapter. They are used for gathering data and transferring it to the FlexNet Manager Suite database. The template adapter provided performs all the differential updates required to move the data into the final FlexNet Manager Suite tables. This script tab provides SQL syntax highlighting, and a special red underlined highlight that shows where you need to modify queries with your own data values.

Figure 7: Red underlined text should be replaced with your own database column and table names.

Log Message and Result panel

Log Message

The log message tab shows the results of executing the adapter. This is the same as the command line logging from the Compliance Importer. Look here for error messages. You can get more detail by setting the verbose tracing option on the toolbar.

Result

This shows the results of highlighting a query in the SQL Script and pressing the **Run Current Script** button. Multiple results sets can be displayed.

TestAdapter - Inventory Adapter Studio
Source Connection: TestConnection (Test)
Version: 1.0
Limit computers to 10
Full Import...
Options...
Step Explorer
Read computer names into FNMP
Create domain tables on the target database
Get domains from source database
Read computer number of processors into FNMP
Step name: Get domains from source database
Execution: SourceToTarget
Destination table: #Domain
File: Domain.xml
Version from:
Up to version:
Step is optional
Runs on full import
Disconnected
SQL Script
Run current script
Run from beginning
Stop
-- The ComplianceImporter will read any domain information from Active Directory as well,
-- so if all your domains are available from there then you can delete the SQL in this step
-- and do nothing.
--
-- TODO: Replace the highlighted parts of the SQL below with a database table and
-- fields that return the appropriate domain values.
-- Do not remove the trimming code from the query, it prevents the import from
-- crashing when the data is transferred to the FNMP database.
SELECT
LEFT(RTRIM(LTRIM(d.DN)), 200) AS QualifiedName,
LEFT(RTRIM(LTRIM(d.FlatDName)), 200) AS FlatName
FROM {Domain} AS d
WHERE d.DN IS NOT NULL
AND RTRIM(LTRIM(d.DN)) != ''
AND RTRIM(LTRIM(SUBSTRING(d.DN , 0, 200))) != ''
SELECT * FROM ExternalAsset
Log Message
Result
AssetID ParentAssetID ShortDescription SerialNumber AssetTypeID AssetTag AssetStatusID PurchasePrice
1 Desktop 55041-006-2508076-86132 1 1
2 ltp-jsmith 12541-098-9988076-88232 2 3

4

Installing Inventory Adapter Studio

For installation on the application server, download the installer `Inventory Adapter Studio releaseNumber.zip` from https://flexerasoftware.flexnetoperations.com. You require an account name and password for this site, which were originally sent to your enterprise as part of the order confirmation process.

Run the installer on a computer with FlexNet Manager Suite already installed. The Inventory Adapter Studio looks up the installation directory, and installs itself in the installation folder of FlexNet Manager Suite.

On your inventory beacon, no separate installation is required. Inventory Adapter Studio is installed as part of the installation process for the inventory beacon.

The Inventory Adapter Studio executable: `InventoryAdapterStudio.exe`
Default location (beacon): `C:\Program Files (x86)\Flexera Software\Inventory Beacon\DotNet\bin`
Default location (application server): `C:\Program Files (x86)\Flexera Software\FlexNet Manager Platform\DotNet\bin`
Template file storage: `C:\ProgramData\Flexera Software\Compliance\ImportProcedures\`, followed by:

- `AdapterStudioTemplates` for templates downloaded from the central application server
- `CustomInventory` for the template file for custom adapters in an on-premises installation where you have access to both the source and target databases simultaneously
- `Inventory` for standard adapters supplied with the product. These implement standard integration with other products.
- `ObjectAdapters`, with a subfolder `Reader` for adapters customized on an inventory beacon to run in disconnected mode.

Starting Inventory Adapter Studio

After installation, you can find a shortcut to Inventory Adapter Studio in the Windows Start menu (**All Programs** > **Flexera** > **Inventory Adapter Studio**).

5

Understanding Inventory Adapters

Inventory adapters exist to extract data from one database (the inventory source), transform it as required, and write it into the destination (or target) database.

To understand the work in creating or modifying an adapter, it is helpful to know a little about:

- The Compliance Importer, considered as the framework which runs adapters
- The resulting structural requirements for an inventory adapter
- What is provided in templates to help you quickly build inventory adapters
- The object model (legal database objects and their properties) for saving content into the destination database when your inventory adapter is running on your inventory beacon in disconnected mode (see Inventory Adapter Object Model).

The Architecture of Compliance Importer

Compliance Importer is the framework within which inventory adapters function, and therefore dictates the requirements for each adapter.

The Compliance Importer is the software that executes inventory adapters to import data into FlexNet Manager Suite. It is a generic data import framework, but specific procedures are provided to import inventory data from source databases.

Once data is imported, it is matched to existing information in FlexNet Manager Suite, the Application Recognition Library is applied, and license compliance is calculated.

The overall model of the Compliance Importer is as follows:

- A set of procedures is defined for the import process. Procedures are grouped by their purposes as Readers, Writers and Export procedures.
- Tables are defined as intermediate storage and workspace for data used by each procedure. In most cases these staging tables are shipped as part of the FlexNet Manager Suite database schema.
- The main purpose of readers is to read data from a source database, and use it to populate the staging tables in the operations database. To fulfill that function, readers in *connected mode* may also load data into the source database

to be used as context in their queries. This contextual information should use temporary database tables so that there is no permanent change to the source database. When operating in disconnected mode on an inventory beacon, the readers work in two stages: writing the gathered data to an intermediate package on the inventory beacon for later upload to the central application server; and subsequently loading the intermediate package data into the staging tables.

- Readers may also perform operations on data in the FlexNet Manager Suite or source database, usually to prepare data before returning results.
- Writers update the operations database, using the data in the staging tables to determine the changes to make.
- The Export step extracts data from the FlexNet Manager Suite database into a data warehouse for presentation in reports that track changes over time.

Understanding the function of the Reader procedures is especially important to preparing inventory adapters.

Structure of an Inventory Adapter

Each inventory adapter is a set of reader instructions for the compliance importer. The permitted structure depends on the origin and operational mode for this adapter.

Each adapter runs in one of (up to) three modes:

- "Connected mode", where the adapter has simultaneous access to both the source and target databases (for example, when the source database is accessible from the server running the operations database for FlexNet Manager Suite).

  Note: *For security reasons, connected mode is not available when you are using FlexNet Manager Suite as a cloud service.*

- "Disconnected mode", where you need to install an inventory beacon, either because the source database and target database are on separate networks, or because you are using FlexNet Manager Suite as a cloud service. For more information see Disconnected Mode.
- "Disconnected mode (Tier 1)", where the same operational conditions apply with the adapter running on an inventory beacon, but because the adapter is factory-supplied, security provisions take a different form, discussed below.

The operations that are available when creating a custom adapter depend on the mode in which it runs.

Note: *Adapters engineered by Flexera and provided as standard functionality (sometimes called Tier 1 adapters) may include operations of all types. However, when working on an inventory beacon, you must not edit any Tier 1 adapters. For security reasons, a modified Tier 1 adapter in disconnected mode is automatically failed, and cannot import any inventory. In contrast, when you edit on your central application server, you may customize Tier 1 adapters, as you then take responsibility for your own security arrangements.*

Table 5: Availability of all adapter operation types

Type of operation	Description	Available in modes
Target to source	Executes a database query on the FlexNet Manager Suite database and copies the result to a table in the source database. This data is to provide context for a subsequent query on the source database.	Connected
Source to target	Executes a database query on the source database and copies the result to a table in the FlexNet Manager Suite database. ***Restriction:*** *For fast transfers, the Compliance Importer uses SQL bulk copy operations to move data from one database to another. This means that the query on the source and the table on the target must match exactly in column order and data type.* For disconnected (tier 1) operations, a source to target step is modified so that data is saved to intermediate packages before upload.	Connected Disconnected (Tier 1)
Source to object	Replaces 'source to target' for use in disconnected mode with custom adapters. Instead of writing source data directly to the target database, it is written into an intermediate package format and saved on the inventory beacon. The intermediate packages are uploaded asynchronously to the cloud server, and processed (by default overnight) for import into the operations database. ***Restriction:*** *Since you cannot modify the internal processing, the data in the intermediate packages must map correctly to a standard set of objects and their attributes (see Inventory Adapter Object Model).*	Disconnected
Execute on source	Executes an SQL script on the source database.	Connected Disconnected (switchable for disconnected *only*) Disconnected (Tier 1)
Execute on target	Executes an SQL script on the FlexNet Manager Suite database.	Connected Disconnected (Tier 1)

The three architectures are shown in the diagrams below. From a security perspective, the distinction between the modes is:

- Connected mode applies only for on-premises installations where the security of both databases is entirely in your control.
- Disconnected mode for custom adapters protects the central operations database in the cloud service by disallowing any custom SQL on the target side. This also limits the permissible imports to the standard set of objects and

attributes available in the Inventory Adapter Studio running on an inventory beacon. You can also find an XML file defining those objects and attributes on your inventory beacon at `C:\ProgramData\Flexera Software\Compliance\ImportProcedures\ObjectAdapters\InventoryObjectModel.xml`.

- In disconnected mode, Tier 1 (factory-supplied) adapters are able to use factory-approved custom SQL on the target side. Security is provided by disallowing the slightest revision of any kind to these adapters. If you change anything on Tier 1 inventory adapters for disconnected mode, they will not run, and importing your inventory will completely fail.

Connected mode
Source database
Compliance Importer
Target database
Destination table
Target to Source step
Query
Query
Source to Target step
Destination table
Script
Execute on Source step
Execute on Target step
Script

Disconnected mode (custom adapters)
Source database
Inventory beacon
Compliance Importer
Target database
Query
Source to Object step
Restricted staging tables
Upload
Standard Import
Destination table
Script
Execute on Source step

Disconnected mode (Tier 1 factory-supplier adapters only)
Source database
Inventory beacon
Compliance Importer
Target database
Query
Source to Target step
Destination table
Upload
Script
Execute on Source step
Execute on Target step
Script

Structure of Templates for Inventory Adapters

Templates are provided for inventory adapters, to speed your development effort.

The adapter templates shipped with the Inventory Adapter Studio use the adapter structure as follows:

- Temporary database tables are set up on the source and FlexNet Manager Suite databases.
- Sample queries are written in Source to Target steps. These write to the temporary tables already created.
- To make each query as easy to write as possible, the minimum number of columns is sent in each query. This also documents the minimum requirements for importing the inventory source.
- Areas of the query that require change are enclosed with curly brackets, colored red, and underlined: {Replace this text}.
- As many of the other steps as possible are already completed. They rely on the fact that data has been transferred in the Source to Target steps.

Each step in the templates has comments describing the updates that need to occur. Optional fields are identified, and the adapter will still work if the optional fields are not provided.

At the end of each procedure there is a lot of provided code that performs a differential update into the Imported database tables in the FlexNet Manager Suite database. It is recommended that you do not change this code for your adapter. There may be special cases where this is required, but an error will prevent any data from being imported into FlexNet Manager Suite.

6

Creating a New Adapter

Once completed and published to FlexNet Manager Suite, each adapter may be used to import from multiple databases that have the same structure. In other words, a number of similar connections (to the databases) may reuse the same adapter.

Note: *There is a separate process for editing an existing adapter.*

To create a new adapter:

1. Click the New icon in the toolbar.

   The **Create New Adapter** dialog opens.

Tip: *If the Inventory Adapter Studio cannot locate downloaded templates, it displays a warning message in this dialog. You can download the latest templates using the* ***Update templates*** *button (if necessary, first re-establishing your link to the application server in the inventory beacon interface).*

2. Specify the name for your adapter. It is best practice to choose a name similar to the data source you plan to import from.

   You may not change the directory the new adapter is saved in. This is because FlexNet Manager Suite uses specific directories to separate out-of-the-box and custom adapters. For example, if you are creating this adapter on an inventory beacon, the default path is `C:\Program Files\Flexera Software\Compliance\ImportProcedures\ObjectAdapters`.

Your adapter appears, pre-populated with samples for each available object. You may remove the examples you do not need, and complete the ones required for your adapter.

Tip: *The templates in the new adapter depend on the context in which you are working. For example, if you created this adapter on an inventory beacon, only* `Source to Object` *steps and* `Execute on Source` *steps are available.*

After you create a new adapter, you must create a new database connection that matches the type of the adapter.

7

Editing an Existing Adapter or Template

You may edit an existing, custom adapter that was created in your enterprise. In disconnected mode (that is, using the cloud service solution), do not attempt to edit any factory-supplied (Tier 1) adapters. If you edit any part of a Tier 1 adapter, it ceases to operate.

To edit an existing adapter or template:

1. Click the Open icon in the toolbar.

   The **Open Existing Adapter** dialog appears. The meaning of the **Type** column is as follows:

   - Adapters of type `BuiltIn` are factory-supplied adapters that implement standard connectivity. You may read but not edit these adapters on an inventory beacon, but you may add customizations if you edit these on your central application server.
   - Adapters you have previously edited on your application server are marked `Custom`. If you are now working on an inventory beacon, these customizations have been automatically replicated from the application server to your inventory beacon.
   - Adapters of type `Object` are those which you have previously edited while working on your inventory beacon.

   On an inventory beacon, you can only open `Object` adapters, stored (by default) in `C:\Program Files\Flexera Software\Compliance\ImportProcedures\ObjectAdapters`.

2. Select the desired custom adapter from the list, and click **OK**.

   Details of the adapter appear in the **Step Explorer** and edit panel.

8

To Create a Source Connection

You must know either the server name or its IP address (together with database instance name, if any) to type in during the following process. (There is no browse facility to find the server.)

This process establishes the link between the adapter and the source inventory database. This connection may be used for both reading inventory, and also writing data (if additional context is required for good information gathering).

This process assumes that you already have the appropriate adapter open in the Step Explorer and edit panel. This process creates a test connection, because new adapters are not ready for production. Test connections are not imported by the Compliance Importer in its normal operations. Data from this connection is imported only after you publish the completed and tested connector.

To create a source connection:

1. In the toolbar, on right-hand end of the **Source Connection** control, click the [**...**] ellipsis button.

   The **Select Source Connection** dialog opens.

Tip: *You must create a new test connection for this adapter. You may not reuse an existing connection for this adapter. (The existing connections are those previous declared on your inventory beacon, and editing or deleting from this dialog affects the connections for your inventory adapter.) Only test connections, those displaying a check mark in the* ***Test*** *column, may be created from the Inventory Adapter Studio.*

2. Click **New...**.

   The **Create SQL Server Connection** dialog opens. (If not, see note below.)

3. Complete the details:

a. Provide a descriptive name in the **Connection Name** field, perhaps referencing the name of the adapter using this connection.

b. From the **Type** pull-down, select the *name of the adapter* you are editing (or just created). Scroll down the list to find your adapter's name. Every connection must be tightly coupled to an adapter through this **Type** setting.

c. In the **Server** field, type the server name or IP address. If the server hosts multiple SQL instances, you may append the appropriate instance name after a backslash. For example, with an instance called `Inst1232`, you could enter `10.200.3.102\Inst1232`.

d. In the **Authentication** field, select the authentication method:

   Windows Authentication — Uses standard Windows authentication to access the server. The credentials of the operator currently logged on will be used to access the SQL Server database. Any operators that require access to the database must be added to the security groups that already have access to the database.

   SQL Authentication — If you select this option, you must then specify an account and password already known to SQL Server. This account's credentials will be used to access the source database, regardless of the operator or account running the adapter.

e. If you selected **SQL Authentication**, complete the **Username** and **Password** fields with the account name and password to be used during SQL authentication.

f. In the **Database** field, type the name of the database, or use the pull-down list to select from database names automatically detected on your specified server.

g. Click **Test Connection**. If the compliance server can successfully connect to the nominated database using the server and authentication details supplied, a `Database connection succeeded` message displays. Click **OK** to close the message. Click **Save** to complete the addition of these connection details.

You cannot save the connection details if the connection test fails. If you cannot get the connection test to succeed, click **Cancel** to cancel the addition of these connection details.

Note: *There is a second type of connection that may be created using the* ***New...*** *button on the* ***Select Source Connection*** *dialog. This is used for database connections to non-Microsoft databases. The difference is that a full database connection string must be entered manually for this connection.*

9

Overview: Process for Developing an Inventory Adapter

Here is your mental roadmap through the development process for inventory adapters, with links to the details.

The development process for inventory adapters:

1. Create a new adapter, normally pre-populated with an appropriate set of steps to gather and process data (Creating a New Adapter).
2. Specify a new connection to a data source. Initially this is a test connection during your development phase. (See To Create a Source Connection)
3. Use the **Step Explorer** to:
   - Add a new step to one of the grouping folders (see Adding a New Step to an Inventory Adapter)
   - Remove any steps provided automatically that are not required in your inventory adapter (see Removing a Step from an Inventory Adapter)
   - Change the execution order of steps within your inventory adapter (see Reordering Steps in an Inventory Adapter)
   - Update the details included within an individual step.

   Note: *You cannot add, delete, or reorder folders in the* ***Step Explorer****. These are optimized for the order of insertion into the operations database. You may only modify or reorder the steps within a given folder.*
4. Test each step in your adapter as you develop it (see Tips for Editing an Adapter and Testing an Adapter). Cycle through until you have created and tested all the steps needed to complete your inventory adapter.
5. Run the entire adapter, and validate that the collected data is as you expect (again, see Tips for Editing an Adapter). On an inventory beacon, validation means unzipping the intermediate package and examining the XML file it contains. Look for your created intermediate package in `C:\Program Data\Flexera Software\Beacon\IntermediateData`.
6. Put the completed and test inventory adapter into production (see Publishing Your Adapter).

Adding a New Step to an Inventory Adapter

You may add customized steps to your inventory adapter, choosing its position in the appropriate folder. (Remember that you cannot edit a factory-supplied inventory adapter on an inventory beacon. To edit a factory-supplier adapter, you must be working on your application server.)

To add a new step to an inventory adapter:

1. In the **Step Explorer**, select one of the following:
   - An empty folder (this has no expander icon to the left of the folder name)
   - In an expanded folder of steps, the current step *after* which you want to insert a new step.

   Tip: *To insert a new step as the first in a folder containing existing steps, insert it as the second step and then move it up the list with the up-arrow icon above the list of steps.*

2. Use the down arrow next to the add icon to expand the choices, and click either:
   - `Custom SQL` to write any SQL scripting that runs on either the source database or your target operations database. Use these steps to massage data within the database, such as scrubbing data into a temporary table within the same database.
   - `Data Transfer` to move data from one database to another. These steps can include custom SQL statements to select the data for transfer.

   The new step appears in the **Step Explorer**, and its details appear in the editing pane on the right, potentially in a new tab (if you are already editing other steps).

3. In the editing pane, change the default value in the **Step name** field to something meaningful that will assist with future maintenance of this adapter.
4. Complete the details of your new step in the editing pane. For more about the fields in the editing pane and the permitted values, see Edit panel.

Removing a Step from an Inventory Adapter

Templates for custom inventory adapters may include sample steps that you don't require. When planning removal of any steps from your custom inventory adapter, remember to consider the potential impact on subsequent steps. There is no undo available for deleting a step.

To remove a step from an inventory adapter:

1. In the Step Explorer, select the step you want to remove. If this step has previously been operational, review its contents to check for possible flow-on effects from its removal.

   Tip: *Do not select a folder, as you cannot delete the folders. You may remove all the steps contained within a folder if need be; but the folders must remain. The delete icon is disabled when you select a folder.*

2. Click the delete icon ( ) at the top of the **Step Explorer**.

   A confirmation dialog appears. Remember that there is no undo available for this delete action.

3. Click **Yes** to proceed with the removal of the step (or **No** to reconsider).

Reordering Steps in an Inventory Adapter

An inventory adapter executes in the order shown in the Step Explorer, from top to bottom. Therefore to change the execution order of the step in your adapter, simply change the display order.

To reorder the steps in an inventory adapter:

1. In the **Step Explorer**, select the step you want to move.
2. Use the up and down icons at the top of the **Step Explorer** to move the step within its folder.

   Tip: *You cannot move a step outside the boundaries of its folder; and you cannot reorder the folders themselves.*

Remember to save your changes with the save icon in the toolbar of the Inventory Adapter Studio.

10

Disconnected Mode

Whenever there is a network discontinuity between the source and target databases, your inventory adapter must function in disconnected mode.

When an inventory adapter runs on your central application server in your FlexNet Manager Suite implementation, it can have free and simultaneous access both to the downstream source database (from which to collect inventory) and to the upstream target database (to which to upload the gathered inventory). If the source database can be directly contacted from the *application server*, there is no need for a disconnected mode. However, there are times when this free access is not available: for example, whenever the source inventory database is on a network separate from the application server.

In these cases, the inventory adapter must be scheduled to execute on an *inventory beacon* in *disconnected mode*.

In practical terms, this means recognizing that certain steps are automatically skipped when the inventory beacon is in disconnected mode, and that alternative steps can be planned to run only in this disconnected mode to ensure that the gathered inventory is still a valid dataset.

Selecting a Step for Connected or Disconnected Modes

Procedural steps in an inventory adapter can be selected to run only in disconnected mode, while others are automatically disallowed in that mode.

All procedural steps of the types `TargetToSource` or `ExecuteOnTarget` are automatically disallowed in disconnected mode (such as when the adapter runs on an inventory beacon). Steps of all other types by default run in both connected and disconnected modes. Use this procedure to flag an individual step to run exclusively in disconnected mode, when it functions as an alternative to disallowed steps.

Tip: *Only `TargetToSource` and `ExecuteOnTarget` steps are disallowed for disconnected mode. It is not possible to make steps of other types apply exclusively to connected mode. Either they apply in both connected and disconnected modes, or you can use this procedure to limit them to disconnected mode only.*

To select a step for connected or disconnected modes:

1. In the Step Explorer, select the step you want to restrict to disconnected mode, so that its details appear in the edit panel.
2. In the properties area at the top of the edit panel, select the **Disconnected** check box.

While this check box is selected, this step runs only when the adapter is executed in disconnected mode, such as on an inventory beacon. This setting is irrelevant when the **Execution** property is set to `TargettoSource` or `ExecuteOnTarget`, as all such steps are disabled for disconnected mode.

Why Special Steps Are Required for Disconnected Mode

Some examples help to clarify the reason for, and the workings of, steps that are executed only in disconnected mode.

The basic upload of data from the inventory beacon to the operations database is not disrupted by disconnected mode: the inventory beacon has special services to ensure than gathered inventory is uploaded at appropriate times.

However, there are other common scenarios for inventory adapters that *are* disrupted. Consider this algorithm, designed to optimize data gathering and upload in connected mode by only collecting differential inventory (inventory that has changed since last collection):

Logical step described	Step type
Create a temporary table `#KnownComputer` on the source, to hold IDs of previously inventoried computers and last known updated date.	`ExecuteOnSource`
Send the result of a `SELECT` statement on the target database, listing previously inventoried computer IDs and last known updated date, into the temporary table `#KnownComputer` on the source database.	`TargetToSource`
Source queries join against the temporary table `#KnownComputer` to optimise their returned results.	`ExecuteOnSource`

This works well in connected mode, and only new/changed inventory records are uploaded when this sequence is executed.

However, in disconnected mode, all `TargetToSource` steps are disabled, leaving the `#KnownComputer` temporary table empty. Following steps that attempt joins against the empty table fail, and no inventory is returned.

We therefore need an alternative step, available in disconnected mode only, to prevent the failure and allow reuse of all the other procedure steps so that inventory is returned. In disconnected mode, it is not possible to select content from the target database; but we can take a different action to prevent the temporary table sitting empty.

Logical step described	Step type	Disconnected	Runs in which modes
Create a temporary table `#KnownComputer` on the source, to hold IDs of previously inventoried computers and last known updated date.	`ExecuteOnSource`	Check box clear	Connected/ Disconnected
Send the result of a `SELECT` statement on the target database, listing previously inventoried computer IDs and last known updated date, into the temporary table `#KnownComputer` on the source database.	`TargetToSource`	Check box ignored	Connected only
Enforce full import by filling in `#KnownComputer` with all current IDs in the source system.	`ExecuteOnSource`	Check box set	Disconnected only
Source queries join against the temporary table `#KnownComputer` to optimise their returned results.	`ExecuteOnSource`	Check box clear	Connected/ Disconnected

We can see that the single extra step allows us to use the adapter in both connected and disconnected modes. In connected mode, it performs a differential inventory. In disconnected mode where differential inventory is not possible, it substitutes a full inventory.

11

Tips for Editing an Adapter

The following principles are helpful when you are developing and testing your inventory adapter. If you need information about the fields in the editing pane and the permitted values, see Edit panel.

Minimize processing times

During development, use the setting to limit the number of computers for test imports. This will potentially save hours of processing while testing your adapter. The control applies a filter to the number of computers that the adapter reads from the source database, allowing validation of your work without requiring that all the data is read and processed.

Start with a template

Start with one of the supplied templates and customize it to suit your data source. Focus on the areas needing change:

- In the Step Explorer, each step that requires editing is shown in bold text. The bold is automatically removed when all areas requiring change have been modified.
- When you have selected a step so that its details are shown in the edit panel, the individual edits required are shown within curly braces, underlined, and in red.
- Every step that needs editing has extensive comments on the data structures and requirements provided in the step details. Following these guidelines will provide the quickest path to a working adapter.

Test each SQL step

As you modify each step, you can test the SQL and inspect the data set it produces, by highlighting the section of your SQL to test, and clicking the **Run current script** button. The result is shown in the **Result** tab below.

Tip: *The **Run current script** button will run the entire step if there is no selection in the SQL script.*

Note: *If the selection (or, when there is no selection, the step) includes any red, underlined text that still requires customization, this produces a syntax error when run.*

Figure 8: This step still requires customization, but the customizable text is not included in the selection, which can safely be run to inspect the results of the individual statement.

Test progressively

As you complete each step, click the **Run from beginning** button. This executes all the steps in the adapter from the start up to and including the one currently being edited. Errors are shown in the **Log Message** tab. In the Step Explorer, steps which succeed are marked with a check mark (tick) and those that fail show a red warning symbol.

Tip: *The adapter is executed in the appropriate mode. For example, when you are developing the adapter on an inventory beacon, it must run in disconnected mode, meaning that all* `Target to Source` *steps are skipped, and all steps with the* ***Disconnected*** *check box set are exercised.*

Note: *If any steps between the start and your present position are still bold (require editing to customize them), they will produce a syntax error on execution.*

For repeated testing of an adapter collecting differential inventory, keep in mind that second and subsequent passes will not collect any results until there is a new inventory collected on a later date for some machine(s). In other words, as you continue testing on any given day, you can expect diminishing returns on any differential inventory collections.

Final test

Finally, when you are ready to run an end-to-end test of your adapter, use the **Full Import** toolbar button.

When you are working on an inventory beacon (*disconnected mode*), in the **Run Full Import** dialog, the **Run readers** check box is set, and the **Run writers** check box is cleared, and both are disabled. This indicates that your full import gathers the data from your source databases to the intermediate package; but that the stage of writing data to the operations database is completely asynchronous, and cannot be controlled from the beacon. As long as the connection for this adapter is in test mode, the inventory data it gathers is never visible in the FlexNet Manager Suite compliance console.

Working on your application server gives you the option of executing the writers as well as the readers in the Compliance Importer. When the writers are run successfully, you can look for imported data in the FlexNet Manager Suite console. (You cannot, however, see the impact on compliance of this latest inventory until the next compliance calculation is run.)

Tip: *While the adapter is marked as a test adapter, only the writers for this individual adapter are run. This provides much faster validation of your work than when you edit a production connector: production connectors require that all writers system-wide are exercised together.*

12

To Save an Adapter

Save early, save often.

To save an adapter:

1. Your adapter is saved every time you do any one of the following:
   a. Click the Save icon in the toolbar.
   b. Use the `Ctrl-S` keyboard shortcut.
   c. Click the **Run from beginning** button to test the adapter.

      All the files are saved first, because running the adapter uses the Compliance Importer to read the files from disk. Anything not saved is not tested by the **Run from beginning** button.

13

Testing an Adapter

Completely validating the operation of an inventory adapter requires checking two stages: the reader and the writers.

Once you have finished updating all the steps that require customization in your new adapter (so that nothing is left shown bold in the Step Explorer), it is time to test that it functions correctly. This can proceed in two stages:

- Executing the adapter from the **Run from beginning** button performs the first half of the import, reading data into staging tables in the FlexNet Manager Suite database.
- To validate the second stage, you need to perform a full import.

These two stages are described in the following tasks.

To Run a Full Import

Only a full import causes the Inventory Adapter Studio to work through the entire process for inventory import. You should attempt a full import only after every step in your adapter has been individually tested. Keep in mind that importing large scale data incorrectly can create a significant workload to back out all the incorrect data! Consider using the **Limit computers to** filter for the early testing, even of the full import process.

To run a full import:

1. Inspect the Step Explorer to validate that no step names are displayed in bold text (still awaiting customization).

   If there are bold steps in the Explorer, you cannot run a full import. Instead, circle back and complete the required customization.

2. In the toolbar, click **Full Import...**.

   The **Run Full Import** dialog appears.

3. For a full import, ensure that the **Run readers** and **Run writers** check boxes are both selected.

   Readers collect data from the source database and write it into staging tables in the FlexNet Manager Suite database. Writers massage the data in the staging tables and transfer it into the operations database. You may use these check boxes to test each stage independently when required.

4. Click **Run Full Import** within the dialog.

The logging from the Compliance Importer is echoed in the dialog.

When the full import is finished, inventory gathered by your adapter has been written into staging tables and (provided that you also ran the writers) into the operations database.

To Diagnose Readers for Your Adapter

Because the inventory adapter is running on your application server, you can inspect the database tables directly to check operation.
Follow this procedure using SQL Server Management Studio on your operations database.

To diagnose readers for your adapter:

1. Look in the `ComplianceConnection` database table to find the `ComplianceConnectionID` used for your adapter (search for the name you gave your adapter).

   The `ComplianceConnectionID` is used as a key in all the staging tables of imported data.

2. Determine which staging tables you want to examine for imported data. The staging tables populated by the default templates are:

   - `ImportedDomain`
   - `ImportedComputer`
   - `ImportedUser`
   - `ImportedFileEvidence`
   - `ImportedInstalledFileEvidence`
   - `ImportedInstalledFileEvidenceUsage`
   - `ImportedInstallerEvidence`
   - `ImportedInstalledInstallerEvidence`
   - `ImportedInstalledWMIEvidence`.

3. Write SQL queries to check that the data you are importing appears in these tables. For example:

```
SELECT * FROM ImportedComputer WHERE ComplianceConnectionID = [the ID of your
   connection]
```

Diagnosing Writers for Your Adapter

Because your adapter runs on your central compliance server, you can inspect the database directly to see the results of your inventory import. You can combine this with the other techniques described here.
The simplest way to validate that your data is being imported all the way into the operations database is to inspect the results in the web interface.

To diagnose writers for your adapter:

1. To ensure that the uploaded data is processed from the staging tables into the operations database, do either of the following:
   - Wait until the next scheduled inventory import. By default, the inventory import and recalculation is triggered overnight.
   - Trigger an inventory import/recalculation now. Be aware that this processes all current data, and is not restricted to your new inventory import. As a result, it may take some time (hours, for a large computer estate). Use the following steps:
     a. In your compliance browser, navigate to the **Reconcile** page (**License Compliance > Reconcile**).

     Restriction: *You must be in a role with administrator rights to be able to include your new inventory import in the reconciliation you run manually.*

     b. Select the **Update inventory for reconciliation** check box.

     This setting ensures that uploaded content is incorporated into the operations database and used for the compliance recalculation. Wait for the import and reconciliation to succeed (monitor the **Last successful reconcile** display on the right of the title bar, refreshing your browser page as necessary).
2. When the reconciliation is complete, examine your imported information, for example in the following locations:
   - The **Discovery & Inventory > All Inventory** page (in the `Inventory` group) shows you all the computers that are being imported, as well as the hardware properties that you have set (remember to check the column chooser). Consider filtering by **Created** date to isolate your new imports.
   - The **Enterprise > All Users** page shows all the users you have imported and the attributes that have been set.
   - The **License Compliance > All Evidence** page has separate tabs to show the installation evidence, file evidence, and access evidence (for application virtualization) that was imported.
   - The **License Compliance > All Applications** page shows all the application installations identified as a result of the evidence import. If your inventory revealed an application for the first time, check for a status of `Unmanaged`.

     Note: *If imported evidence did not match any existing application rules, the application does not show in any application list. It will appear only when the Application Recognition Library is updated with new rules incorporating your new evidence.*
   - Usage data is not easily visible in the user interface (usage displays on licenses, but for this validation you need to see it on applications as well). The following query shows the applications on computers that have usage data recorded:

```
SELECT st.FullName AS [Application Name],
                      c.ComputerName,
                      d.QualifiedName AS UserDomain,
                      u.SAMAccountName,
                      usage.UsageSessions,
```

```
                        usage.UsageActiveTime,
                        usage.LastUsedDate
                        FROM  dbo.InstalledSoftware
                        AS isw
                        JOIN dbo.InstalledSoftwareUsage AS usage
                        ON usage.ComplianceComputerID =
isw.ComplianceComputerID AND usage.SoftwareTitleID = isw.SoftwareTitleID
                        JOIN dbo.SoftwareTitle AS st
                        ON st.SoftwareTitleID = isw.SoftwareTitleID
                        JOIN dbo.ComplianceComputer AS c
                        ON c.ComplianceComputerID = isw.ComplianceComputerID
                        LEFT OUTER JOIN dbo.ComplianceUser AS u
                        ON u.ComplianceUserID = usage.ComplianceUserID
                        LEFT OUTER JOIN dbo.ComplianceDomain AS d
                        ON u.ComplianceDomainID = d.ComplianceDomainID
```

14

Publishing Your Adapter

Publishing an inventory adapter means taking it out of test mode and putting it into production. The data uploaded by a published adapter goes into your production database. Be sure you have adequately validated the information gathered by your adapter before taking this step. On an inventory beacon, validation includes unzipping the archive package, saved at `C:\ProgramData\Flexera Software\Beacon\IntermediateData`, and examining the data contained in the XML file. When you are satisfied with the gathered data, you can publish your adapter into production.

To publish your adapter:

1. In the toolbar, on right-hand end of the **Source Connection** control, click the [**...**] ellipsis button.
2. In the **Select Source Connection** dialog, ensure that the correct connection is selected, and click **Edit...**.
3. In the **Create SQL Server Connection** dialog, clear the **Is Test Connection** check box.

4. Click **Save**.
5. Click **OK**.

   This connection is now visible in the connections dialog on the application server. The Compliance Importer can now use this connection with your adapter for future inventory imports. You declare and choose a schedule for execution of this connection in the inventory beacon user interface.

15

Inventory Adapter Object Model

A reference for all database objects, and their properties, that can be imported through your inventory beacon in disconnected mode.

Here is a complete list of the database objects (and their permissible attributes) that you may import through a custom inventory adapter that runs on your inventory beacon.

Inventory Object: AccessingDevice

`AccessingDevice` objects are uploaded to the `ImportedAccessingDevice` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedAccessingDevice` table holds a record client access device information.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccessingDeviceID`	A numeric reference into a static table. May be null.	Matching accessing device ID. Foreign key to the `AccessingDevice` table.
`ComputerName`	Alpha-numeric text (maximum 256 characters). May be null.	Computer name of the client accessing device.
`Domain`	Alpha-numeric text (maximum 100 characters). May be null.	Domain name of the client accessing device.
`ExternalAccessing DeviceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used to identify the device in source connection
`IPAddress`	An ASCII string of alphanumeric characters and punctuation (length 256 characters). May be null.	IP Address of the client accessing device.
`SerialNo`	Alpha-numeric text (maximum 100 characters). May be null.	Serial no of the client accessing device.

Inventory Object: AccessingUser

`AccessingUser` objects are uploaded to the `ImportedAccessingUser` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedAccessingUser` table holds a record of the user access infomarion.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccessingUserID`	A numeric reference into a static table. May be null.	The matchihng AccessingUser ID. Foreign key to the `AccessingUser` table.
`DomainName`	Alpha-numeric text (maximum 100 characters). May be null.	Domain name of the accessing user.
`ExternalAccessing UserID`	Unsigned integer (`bigint`). Mandatory. Database key.	The accessing user id. This is part of the key.
`SAMAccountName`	Alpha-numeric text (maximum 64 characters). May be null.	SAM account name of the accessing user.
`UserName`	Alpha-numeric text (maximum 256 characters).	User name of the accessing user.

Inventory Object: ActiveDirectoryComputer

`ActiveDirectoryComputer` objects are uploaded to the `ImportedActiveDirectoryComputer` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedActiveDirectoryComputer` table stores the incoming active directory data for computers.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ComputerName`	Alpha-numeric text (maximum 64 characters).	The name of the computer. In Windows, this is the NetBIOS name of the local computer, as returned by `GetComputerName()`. For UNIX, it is the host name of the machine, as returned by `gethostname(2)`.
`DomainName`	Alpha-numeric text (maximum 100 characters).	The domain name for the computer.
`GUID`	A universally unique identifier. Mandatory. Database key.	The GUID of the computer.

Property	Attributes	Notes
SID	Alpha-numeric text (maximum 256 characters). May be null.	The SID of the computer.

Inventory Object: ActiveDirectoryDomain

ActiveDirectoryDomain objects are uploaded to the ImportedActiveDirectoryDomain table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedActiveDirectoryDomain table stores the incoming active directory domains for a connection source.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
DomainFQDN	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	The fully qualified name domain name of the AD domain
FlatName	Alpha-numeric text (maximum 32 characters).	The AD domain flat name
LastADImportTime	Date/time field.	The last time the AD data was imported

Inventory Object: ActiveDirectoryExternalMember

ActiveDirectoryExternalMember objects are uploaded to the ImportedActiveDirectoryExternalMember table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedActiveDirectoryExternalMember table stores the incoming active directory data for external AD member objects.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ParentGroupGUID	A universally unique identifier. Mandatory. Database key.	The parent AD group GUID.
SID	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The SID of the member object.

Inventory Object: ActiveDirectoryGroup

`ActiveDirectoryGroup` objects are uploaded to the `ImportedActiveDirectoryGroup` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedActiveDirectoryGroup` table stores the incoming active directory data for a connection source.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`DomainName`	Alpha-numeric text (maximum 100 characters).	The domain name for the user.
`GUID`	A universally unique identifier. Mandatory. Database key.	The GUID of the AD group.
`Name`	Alpha-numeric text (maximum 128 characters). May be null.	The AD group name
`SID`	Alpha-numeric text (maximum 256 characters). May be null.	The SID of the AD group.

Inventory Object: ActiveDirectoryMember

`ActiveDirectoryMember` objects are uploaded to the `ImportedActiveDirectoryMember` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedActiveDirectoryMember` table stores the incoming active directory data for AD member objects.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`GUID`	A universally unique identifier. Mandatory. Database key.	The GUID of the member object.
`ParentGroupGUID`	A universally unique identifier. Mandatory. Database key.	The parent AD group GUID.

Inventory Object: ActiveDirectoryUser

`ActiveDirectoryUser` objects are uploaded to the `ImportedActiveDirectoryUser` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedActiveDirectoryUser` table stores the incoming active directory data for users.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`DomainName`	Alpha-numeric text (maximum 100 characters).	The domain name for the user.
`GUID`	A universally unique identifier. Mandatory. Database key.	The GUID of the user.
`SAMAccountName`	Alpha-numeric text (maximum 20 characters).	The user name.
`Sid`	Alpha-numeric text (maximum 256 characters). May be null.	The Sid for the user.

Inventory Object: ActiveSyncDevice

`ActiveSyncDevice` objects are uploaded to the `ImportedActiveSyncDevice` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedActiveSyncDevice` table stores details of ActiveSync partnerships. A partnership is a user/device pair, so there may be multiple rows for one device.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ActiveSyncID`	Alpha-numeric text (maximum 512 characters). Mandatory. Database key.	The EASIdentity presented by the source, a combination of the AD user and the unique device ID. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
DeviceID	Alpha-numeric text (maximum 100 characters). May be null.	The unique device identifier.
DeviceModel	Alpha-numeric text (maximum 100 characters). May be null.	The device model.
DeviceOS	Alpha-numeric text (maximum 100 characters). May be null.	The device operating system.
DeviceType	Alpha-numeric text (maximum 50 characters). May be null.	The device type.
DeviceUserAgent	Alpha-numeric text (maximum 100 characters). May be null.	The device user agent; an ActiveSync client-specific value that may identify the device type.
Domain	Alpha-numeric text (maximum 100 characters). May be null.	The domain of the device. This may be a flat name or FQDN.
EmailAddress	Alpha-numeric text (maximum 256 characters). May be null.	The user's primary email address.
ExchangeServer	Alpha-numeric text (maximum 256 characters). May be null.	The source exchange server for this information.
IMEI	Alpha-numeric text (maximum 256 characters). May be null.	IMEI (International Mobile Equipment Identity) is a 15- or 17-digit code that uniquely identifies mobile phone sets. Leave blank (null) for other device types.
LastSuccessSync	Date/time field. May be null.	The last successful sync time for this partnership, in UTC.
LastSyncAttemptTime	Date/time field. May be null.	The last attempted sync time for this partnership, in UTC.
PhoneNumber	Alpha-numeric text (maximum 128 characters). May be null.	The phone number of the device. Used for mobile devices.
UserDisplayName	Alpha-numeric text (maximum 256 characters). May be null.	The AD user display name.
WhenCreatedUTC	Date/time field. May be null.	The date/time this partnership was created, in UTC.

Inventory Object: ClientAccessEvidence

`ClientAccessEvidence` objects are uploaded to the `ImportedClientAccessEvidence` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedClientAccessEvidence` table holds all of the client access evidence which has been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`Edition`	Alpha-numeric text (maximum 50 characters). May be null.	The edition of the installed product.
`ExternalAccess EvidenceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier of the client access evidence.
`ProductName`	Alpha-numeric text (maximum 256 characters). May be null.	The name of the product being accessed by user or computer. This may include version and edition too.
`UALRoleGUID`	A universally unique identifier. May be null.	The UAL role GUID of the product being accessed by user or computer. This is used when retrieve data using UAL.
`UALRoleName`	Alpha-numeric text (maximum 256 characters). May be null.	The UAL role name of the product being accessed by user or computer. This is used when retrieve data using UAL.
`Version`	Alpha-numeric text (maximum 72 characters). May be null.	The version of the installed product.

Inventory Object: ClientAccessEvidenceMapping

`ClientAccessEvidenceMapping` objects are uploaded to the `ImportedClientAccessEvidenceMapping` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedClientAccessEvidenceMapping` is the mapping table for imported access evidence and access evidence

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
AccessEvidenceID	A numeric reference into a static table. Mandatory. Database key.	Access evidend id. Foreign key to AccessEvidence table.
ExternalAccess EvidenceID	Unsigned integer (bigint). Mandatory. Database key.	External Access evidend id. Foreign key to ImportedClientAccessedAccessEvidence table.

Inventory Object: ClientAccessedAccessEvidence

ClientAccessedAccessEvidence objects are uploaded to the ImportedClientAccessedAccessEvidence table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedClientAccessedAccessEvidence table holds a record of the installer evidence that has been installed on a computer from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ClientAccessSource	Alpha-numeric text (maximum 100 characters). Default: 'Unknown'. Mandatory. Database key.	Referencing to the client access source type.
ExternalAccess EvidenceID	Unsigned integer (bigint). Mandatory. Database key.	Access evidence id .Foreign key to the ImportedClientAccessEvidence table.
ExternalAccessing DeviceID	Unsigned integer (bigint). Mandatory. Database key.	Accessing computer id .Foreign key to the ImportedAccessingDevice table ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`ExternalAccessing UserID`	Unsigned integer (`bigint`). Mandatory. Database key.	Accessing userid. Foreign key to the `ImportedAccessingUser` table ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalServer ComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	Server computer id .Foreign key to the `ImportedComputer` table. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ImportedClient AccessedAccess EvidenceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the installer evidence.

Inventory Object: ClientAccessedAccessOccurrence

`ClientAccessedAccessOccurrence` objects are uploaded to the `ImportedClientAccessedAccessOccurrence` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedClientAccessedAccessOccurrence` table holds the access information of device or user

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
AccessCount	Unsigned integer (int). Default: 1.	Number of access frequency for given date
AccessDate	Date/time field. May be null.	The access date.
ImportedClient AccessedAccess EvidenceID	Unsigned integer (bigint). Mandatory. Database key.	Access evidence id. Foreign key to the ImportedClientAccessedAccessEvidence table.
InventoryDate	Date/time field.	Date on which inventory occurance was recorded.
LicenseDate	Date/time field. Mandatory. Database key.	Date which will be used for licensing purpose.

Inventory Object: Cluster

Cluster objects are uploaded to the ImportedCluster table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedCluster table holds all of the clusters which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ClusterTypeID	A numeric reference into a static table. Default: 1.	The type of cluster.
DPM	Boolean (0 or 1). May be null.	Whether Distributed Power Management (DPM) is enabled
DRS	Boolean (0 or 1). May be null.	Whether Distributed Resource Scheduler (DRS) is enabled
ExternalID	Unsigned integer (bigint). Mandatory. Database key.	The unique identifier for this imported cluster. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
ExternalName	Alpha-numeric text (maximum 256 characters). May be null.	The identifier of the cluster in the external cluster management system.

Property	Attributes	Notes
InventoryAgent	Alpha-numeric text (maximum 64 characters). Default: ''. May be null.	The name of the person or tool that performed the last inventory.
InventoryDate	Date/time field. May be null.	The date the cluster last had inventory reported.
Name	Alpha-numeric text (maximum 256 characters).	The user-visible name of the cluster.
Namespace	Alpha-numeric text (maximum 256 characters). May be null.	The name of the domain/datacenter containing the cluster.

Inventory Object: ClusterGroup

ClusterGroup objects are uploaded to the ImportedClusterGroup table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedClusterGroup table holds all of the group objects defined on clusters which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ClusterExternalID	Unsigned integer (bigint). Mandatory. Database key.	The unique identifier for the imported cluster.
ClusterID	A numeric reference into a static table. May be null.	The assigned identifier for this cluster group.
ClusterTypeID	A numeric reference into a static table. Default: 3.	Foreign key to the ClusterType table.
ExternalID	Unsigned integer (bigint). Mandatory. Database key.	The unique identifier for this imported cluster group. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
Name	Alpha-numeric text (maximum 256 characters).	The name of the cluster group.

Inventory Object: ClusterGroupMember

`ClusterGroupMember` objects are uploaded to the `ImportedClusterGroupMember` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedClusterGroupMember` table holds all of the group memberships defined on clusters which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ClusterGroupExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The unique identifier for the imported cluster group.
`ComputerExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the external computer which is a member of the group. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: ClusterHostAffinityRule

`ClusterHostAffinityRule` objects are uploaded to the `ImportedClusterHostAffinityRule` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedClusterHostAffinityRule` table holds all of the host affinity rules for a cluster which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ClusterExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The unique identifier for the imported cluster.
`ClusterHostAffinity RuleTypeID`	A numeric reference into a static table. Default: 1.	A unique identifier indicating a type of Cluster Host Affinity Rule.

Property	Attributes	Notes
`ClusterHostGroup ExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The unique identifier for the imported cluster host group.
`ClusterVMGroup ExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The unique identifier for the imported cluster VM group.
`Name`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The name of the cluster group.

Inventory Object: ClusterNode

`ClusterNode` objects are uploaded to the `ImportedClusterNode` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedClusterNode` table holds all of the cluster nodes which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ClusterExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The unique identifier for the imported cluster.
`ClusterNodeTypeID`	A numeric reference into a static table. Default: 1.	Foreign key to the `ClusterNodeType` table.
`ComputerExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the external computer which is a member of the cluster. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: Computer

`Computer` objects are uploaded to the `ImportedComputer` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedComputer` table holds all of the computers which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`CalculatedUser`	Alpha-numeric text (maximum 128 characters). May be null.	The domain/SAMAccountName of the calculated user. Some inventory systems calculate the user who owns a computer. For example, it might be the user who, over the last ten logins, logged in most often.
`ChassisType`	Alpha-numeric text (maximum 128 characters). May be null.	The type of case of the computer. The value must be a (case insensitive) exact match for one of the values shown. Note that some license types use this information to optimize the licensing position, particularly with desktop and laptop computers.
`ComplianceComputerTypeID`	Unsigned integer (`int`). May be null.	If you know that the computer is a virtual machine or VM host, record that data here. If you are unsure, leave this cell empty (NULL): this allows the system to infer the computer type (for example, a computer with VMs linked to it is inferred to be a VM host). If data comes from multiple inventory sources, leaving this value as null also allows the value to be inserted from another source. So, unless there is a very good reason, do not just specify 'Computer', but allow the inference rules to help.
`ComputerName`	Alpha-numeric text (maximum 256 characters). May be null.	The name of the computer. In Windows, this is the NetBIOS name of the local computer, as returned by `GetComputerName()`. For UNIX, it is the host name of the machine, as returned by `gethostname(2)`.
`Domain`	Alpha-numeric text (maximum 100 characters). May be null.	The domain of the computer.
`EmailAddress`	Alpha-numeric text (maximum 256 characters). May be null.	The email address associated with the device. Typically used for mobile devices.

Property	Attributes	Notes
`ExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`FirmwareSerialNumber`	Alpha-numeric text (maximum 100 characters). May be null.	Serial number in the system firmware such as BIOS, EEPROM etc.
`HardwareInventoryDate`	Date/time field. May be null.	The date (and optionally time) when the hardware was last inventoried. For automated/scheduled data uploads through an inventory beacon, make sure that inventory dates are kept current, as they are used to report out-of-date inventory sources. For a one-time upload to the central application server, leave inventory dates empty (null). At each import from the saved file, the import date is used as the inventory date, which prevents the inventory becoming stale. Notice that this value is not available in the web interface.
`HostID`	Alpha-numeric text (maximum 100 characters). May be null.	The HostID hardware property for the server hosting this machine partition (when inventorying a machine partition such as Solaris Zone, AIX lPar, HP-UX nPar/vPar).
`HostIdentifyingNumber`	Alpha-numeric text (maximum 128 characters). May be null.	Virtual hosts may have an identifier that is unique only across that hardware model. It is less unique than the true hardware serial number, for example.
`HostType`	Alpha-numeric text (maximum 128 characters). May be null.	The type of the physical host computer.

Property	Attributes	Notes
ILMTAgentID	Unsigned integer (bigint). May be null.	The unique ID used by the IBM License Metric Tool (ILMT) inventory agent on this device, if the inventory source is aware of this value. This can be used to track a computer over time and can be used to socialize different inventory sources. Currently the ILMT and ManageSoft inventory adapters report this value.
IMEI	Alpha-numeric text (maximum 256 characters). May be null.	IMEI (International Mobile Equipment Identity) is a 15- or 17-digit code that uniquely identifies mobile phone sets. Leave blank (null) for other device types.
IPAddress	Alpha-numeric text (maximum 256 characters). May be null.	The IP address of the computer.
IgnoredDueToLicense	Boolean (0 or 1). Default: 0.	True if this machine is not imported into compliance computer table due to license limitation
IncompleteRecord	Boolean (0 or 1). May be null.	Used to identify records which do not have all information specified. Primarily used for ManageSoft source connections where the domain name was not reliably reported.
InventoryAgent	Alpha-numeric text (maximum 128 characters).	The name of the person or tool that performed the last inventory. For imported spreadsheets, you may wish to include the name of the person preparing the data, in case there is subsequent follow-up required.
InventoryDate	Date/time field. May be null.	The date the computer last had inventory reported.
IsDuplicate	Boolean (0 or 1). Default: 0.	Used to identify that imported computer is a duplicate of another, whereby a new computer will not created.
IsRemoteACLDevice	Boolean (0 or 1). Default: 0.	Used to determine if the current record is a remote ACL based device.
LastLoggedOnUser	Alpha-numeric text (maximum 128 characters). May be null.	The DOMAIN/SAMAccountName of the user last logged onto the computer.

Property	Attributes	Notes
LastSuccessful InventoryDate	Date/time field. May be null.	For incremental imports, this represents the inventory date of the computer in the source at the time this record was last successfully imported. If the import procedure has failed, this may be different to the inventory date. At the end of a successful incremental import, this value is updated to match the inventory date. If no value is present in this field, either there has not been a successful import of this computer or the reader for this record is not using an incremental update model.
LegacySerialNo	Alpha-numeric text (maximum 100 characters). May be null.	A previous serial number of this computer that can also be used for matching.
MACAddress	Alpha-numeric text (maximum 256 characters). May be null.	The MAC address of the computer.
MDScheduleContainsPV UScan	Boolean (0 or 1). Default: 0. May be null.	Does this managed device include an event in its current schedule for running extra IBM PVU hardware scans.
MDScheduleGenerated Date	Date/time field. May be null.	The last time the managed device schedule was regenerated.
MachineID	Alpha-numeric text (maximum 100 characters). May be null.	For AIX, it is the System ID. For HP-UX, it is the Machine/Software ID. It is unset for other platforms.
Manufacturer	Alpha-numeric text (maximum 128 characters). May be null.	The manufacturer of the computer hardware.
MaxClockSpeed	Unsigned integer (int). May be null.	The maximum clock speed of the fastest processor in the computer.
ModelNo	Alpha-numeric text (maximum 128 characters). May be null.	The model number of the computer.
NumberOfCores	Unsigned integer (int). May be null.	The number of cores in the computer.
NumberOfDisplay Adapters	Unsigned integer (int). May be null.	The number of graphics cards in the computer.
NumberOfHardDrives	Unsigned integer (int). May be null.	The number of hard drives in the computer.
NumberOfLogical Processors	Unsigned integer (int). May be null.	The number of logical processors in the computer.
NumberOfNetworkCards	Unsigned integer (int). May be null.	The number of network cards in the computer.

Property	Attributes	Notes
NumberOfProcessors	Unsigned integer (int). May be null.	The number of processors in the computer.
NumberOfSockets	Unsigned integer (int). May be null.	The number of sockets in the computer.
OperatingSystem	Alpha-numeric text (maximum 128 characters). May be null.	The operating system of the computer.
PartialNumberOf Processors	Fractional number (float). May be null.	The fractional processor count available to this computer.
PhoneNumber	Alpha-numeric text (maximum 128 characters). May be null.	The phone number of the device. Used for mobile devices.
ProcessorType	Alpha-numeric text (maximum 256 characters). May be null.	The type of processor in the computer.
SerialNo	Alpha-numeric text (maximum 100 characters). May be null.	The serial number of the computer.
ServicePack	Alpha-numeric text (maximum 128 characters). May be null.	The service pack installed for the operating system.
ServicesInventoryDate	Date/time field. May be null.	The date when services (for example, Oracle) were last scanned on this computer. For automated/scheduled data uploads through an inventory beacon, make sure that inventory dates are kept current, as they are used to report out-of-date inventory sources. For a one-time upload to the central application server, leave inventory dates empty (null). At each import from the saved file, the import date is used as the inventory date, which prevents the inventory becoming stale.
TotalDiskSpace	Unsigned integer (bigint). May be null.	The total size of all hard drives in the computer.
TotalMemory	Unsigned integer (bigint). May be null.	The total RAM in the computer, in bytes.
UUID	A universally unique identifier. May be null.	The BIOS UUID of the computer.
UntrustedSerialNo	Boolean (0 or 1). Default: 0.	Is this computer known to have a serial number from a data source that should not be trusted.

Inventory Object: ComputerCustomProperty

`ComputerCustomProperty` objects are uploaded to the `ImportedComputerCustomProperty` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedComputerCustomProperty` table is used by the importer to import custom properties for computers.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier, in the source connection, of the computer that this property belongs to.
`PropertyNameID`	Unsigned integer (`int`). Mandatory. Database key.	The identifier for custom property in the `ImportedCustomPropertyName` table.
`PropertyValue`	Alpha-numeric text (maximum 256 characters).	The value of the custom property.

Inventory Object: ConsolidatedAccessEvidence

`ConsolidatedAccessEvidence` objects are uploaded to the `ConsolidatedAccessEvidence` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

ConsolidatedAccessEvidence provides a simpler interface to specify client access happening on application installed on server computers. It combines the server computer, and its access evidence details into a single row.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccessCount`	Unsigned integer (`int`). Default: 1. May be null.	Number of times the product was accessed on the given access date.

Property	Attributes	Notes
`AccessDate`	Date/time field. Mandatory. Database key.	The access date of the access evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`AccessingDeviceComputerName`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	IP Address of the device accessing the product. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`AccessingDeviceDomain`	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	Domain name of the device accessing the product. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`AccessingDeviceIP Address`	An ASCII string of alphanumeric characters and punctuation (length 256 characters). Mandatory. Database key.	IP Address of the accessing device. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`AccessingDevice SerialNo`	Alpha-numeric text (maximum 100 characters). May be null.	Serial number of the device accessing the product.
`AccessingUser`	Alpha-numeric text (maximum 128 characters). Mandatory. Database key.	The DOMAIN/SAMAccountName of the user accessing the product. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ClientAccessSource`	Alpha-numeric text (maximum 100 characters). Default: Manual. May be null.	The source type of the access evidence.
`ComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer. It must match the ComputerID from the Computer spreadsheet or the row will be ignored.

Property	Attributes	Notes
`Edition`	Alpha-numeric text (maximum 50 characters). Default: . Mandatory. Database key.	The edition of the software as reported by the access evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`InventoryDate`	Date/time field. Default: getdate(). May be null.	The date (and optionally time) the access evidence record was inventoried.
`ProductName`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The product name of the software as reported by the access evidence.
`Version`	Alpha-numeric text (maximum 72 characters). Default: . Mandatory. Database key.	The version of the software as reported by the access evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: ConsolidatedCluster

`ConsolidatedCluster` objects are uploaded to the `ConsolidatedCluster` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The Cluster spreadsheet provides a simple interface for defining server clustering. It is useful when combined with the ClusterGroup and ClusterHostAffinityRule spreadsheets.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ClusterID	Unsigned integer (bigint). Mandatory. Database key.	The unique identifier for this imported cluster. This may be a string or an integer.
ClusterName	Alpha-numeric text (maximum 128 characters).	The name of the cluster in the external cluster management system.
ClusterType	Alpha-numeric text (maximum 128 characters).	The kind of cluster. The value must be an exact case-insensitive match to one of the permitted values.
DPM	Boolean (0 or 1). May be null.	Whether Distributed Power Management (DPM) is enabled on the cluster.
DRS	Boolean (0 or 1). May be null.	Whether Distributed Resource Scheduler (DRS) is enabled on the cluster.
InventoryAgent	Alpha-numeric text (maximum 64 characters). May be null.	The name of the person or tool that performed the last inventory. For imported spreadsheets, you may wish to include the name of the person preparing the data, in case there is subsequent follow-up required.
InventoryDate	Date/time field. May be null.	The date (with optional time) that the cluster last had inventory reported.
Namespace	Alpha-numeric text (maximum 256 characters). May be null.	Where the cluster is contained.

Inventory Object: ConsolidatedClusterGroup

ConsolidatedClusterGroup objects are uploaded to the ConsolidatedClusterGroup table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ClusterGroup spreadsheet uses data from the Cluster spreadsheet and defines groups of servers as well as computers that are members of these groups.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ClusterGroupID	Unsigned integer (bigint). Mandatory. Database key.	The unique identifier for this cluster group. This may be a string or an integer.
ClusterGroupName	Alpha-numeric text (maximum 128 characters). May be null.	The name of the cluster group. Depending on the value of the ClusterGroupType this will be a group of hosts or virtual machines.

Property	Attributes	Notes
`ClusterGroupType`	Alpha-numeric text (maximum 128 characters).	The kind of cluster included in the group. The value must be an exact case-insensitive match to one of the permitted values.
`ClusterID`	Unsigned integer (`bigint`). Mandatory. Database key.	The unique identifier for the imported cluster. This may be a string or an integer and must match a value for the ClusterID in the cluster spreadsheet.
`ComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the 'Computer' spreadsheet for a computer which is a member of the group. To identify all the members of the group, repeat as many lines as required in your spreadsheet where the other values in the row are identical, and only the 'ComputerID' value changes. Values in this column must match a ComputerID in the computer spreadsheet or the row will be skipped.

Inventory Object: ConsolidatedClusterHostAffinityRule

`ConsolidatedClusterHostAffinityRule` objects are uploaded to the `ConsolidatedClusterHostAffinityRule` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ClusterHostAffinity spreadsheet defines the groups of virtual machines which may run on groups of host servers.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ClusterHostAffinity RuleType`	Alpha-numeric text (maximum 128 characters).	The type of affinity rule. The value must be an exact case-insensitive match to one of the permitted values.
`ClusterHostGroupName`	Unsigned integer (`bigint`). Mandatory. Database key.	The name of the group of hosts that the ClusterVMGroupName virtual machines may run on.
`ClusterID`	Unsigned integer (`bigint`). Mandatory. Database key.	The unique identifier for the imported cluster, to which this affinity rule applies. This may be a string or an integer and must match a ClusterID from the cluster spreadsheet.

Property	Attributes	Notes
`ClusterVMGroupName`	Unsigned integer (`bigint`). Mandatory. Database key.	The name of the virtual machine group that may run on the ClusterHostGroupName hosts.
`Name`	Alpha-numeric text (maximum 128 characters). May be null.	The name of the cluster host affinity rule.

Inventory Object: ConsolidatedComputer

`ConsolidatedComputer` objects are uploaded to the `ConsolidatedComputer` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

'ConsolidatedComputer' consolidates data for the Computer, VirtualMachine, Domain, User and Cluster objects, providing a simpler way to populate this information. Any spreadsheet row that includes a 'HostComputerID' is making that row a virtual machine, and the import process expects that virtualization data will be provided.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AffinityEnabled`	Boolean (0 or 1). Default: 0.	Set this to `true` (or `1`) if this VM has affinity for its current host (so that it is unable to move to different host computers).
`BIOSUUID`	A universally unique identifier. May be null.	The BIOS UUID of the computer (physical or virtual), as provided by the operating system.
`CPUAffinity`	Alpha-numeric text (maximum 256 characters). May be null.	Contains a comma-separated list of processor numbers (Host Logical Processors) or ranges for which this virtual machine has affinity. Example: `1,3-5,8`
`CPUUsage`	Unsigned integer (`int`). May be null.	The maximum CPU usage of the virtual machine (MHz).
`CalculatedUser`	Alpha-numeric text (maximum 128 characters). May be null.	The domain/SAMAccountName of the calculated user. Some inventory systems calculate the user who owns a computer. For example, it might be the user who, over the last ten logins, logged in most often.
`ChassisType`	Alpha-numeric text (maximum 128 characters). May be null.	The chassis type of the device.

Property	Attributes	Notes
`ClusterID`	Unsigned integer (`bigint`). Mandatory. Database key.	The unique identifier for the cluster containing this computer. This must match the ClusterID used in the Cluster spreadsheet. If both the ClusterID and the ClusterNodeType do not match the data provided in the Cluster spreadsheet then the computer will not be associated with a cluster. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ClusterNodeType`	Alpha-numeric text (maximum 128 characters). Default: 1. May be null.	The Cluster node type of the computer. Must be a (case insentitive) exact match for one of the values shown. If both the ClusterID and the ClusterNodeType do not match the data provided in the Cluster spreadsheet then the computer will not be associated with a cluster.
`ComplianceComputerType`	Alpha-numeric text (maximum 128 characters). May be null.	If you know that the computer is a virtual machine or VM host, record that data here. If you are unsure, leave this cell empty (NULL): this allows the system to infer the computer type (for example, a computer with VMs linked to it is inferred to be a VM host). If data comes from multiple inventory sources, leaving this value as null also allows the value to be inserted from another source. So, unless there is a very good reason, do not just specify 'Computer', but allow the inference rules to help.
`ComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The unique identifier for a computer (either physical or virtual). This identifier can either be an integer or a string. Keep this consistent across multiple imports: it is used to track the computer over time.

Property	Attributes	Notes
`ComputerName`	Alpha-numeric text (maximum 256 characters).	The name of the computer. In Windows, this is the NetBIOS name of the local computer, as returned by `GetComputerName()`. For UNIX, it is the host name of the machine, as returned by `gethostname(2)`.
`CoreAffinity`	Alpha-numeric text (maximum 256 characters). May be null.	Contains a comma-separated list of core numbers (or ranges) for which this virtual machine has affinity. Cores are numbered sequentially up the sequence of processors. Example: `1,5-8,10`
`DomainFlatName`	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	The flatname of the domain of the computer. Example: 'mycompany'. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`DomainQualifiedName`	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	The fully qualified domain name for the computer. Example: 'prod.mycompany.eu'. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`EmailAddress`	Alpha-numeric text (maximum 256 characters). May be null.	The email address associated with the device. Typically used for mobile devices.
`FirmwareSerialNumber`	Alpha-numeric text (maximum 100 characters). May be null.	The Serial number in the system firmware such as BIOS, EEPROM etc.

Property	Attributes	Notes
`HostComputerID`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The ComputerID of the server this virtual machine is hosted on. This may be a string or an integer and must match the ComputerID for another computer in this spreadsheet. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`HostID`	Alpha-numeric text (maximum 100 characters). May be null.	The HostID hardware property for the server hosting this machine partition (when inventorying a machine partition such as Solaris Zone, AIX lPar, HP-UX nPar/vPar).
`HostIdentifyingNumber`	Alpha-numeric text (maximum 128 characters). May be null.	Virtual hosts may have an identifier that is unique only across that hardware model. It is less unique than the true hardware serial number, for example.
`HostType`	Alpha-numeric text (maximum 128 characters). May be null.	The type (similar to model number) of the host, used for matching.
`IMEI`	Alpha-numeric text (maximum 256 characters). May be null.	IMEI (International Mobile Equipment Identity) is a 15- or 17-digit code that uniquely identifies mobile phone sets. Leave blank (null) for other device types.
`IPAddress`	Alpha-numeric text (maximum 256 characters). May be null.	The IP address of the computer in IPv4 or IPv6 format.
`InventoryDate`	Date/time field. Default: getdate(). May be null.	The date (and optionally time) the computer last had inventory reported. This field is generally used for differential updates (that is, if the date/time has not changed since the previous import, the data record is not imported/updated).

Property	Attributes	Notes
`LastLoggedOnUser`	Alpha-numeric text (maximum 128 characters). Mandatory. Database key.	The DOMAIN/SAMAccountName of the user last logged onto the computer. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`LastLogonDate`	Date/time field. May be null.	The date and time when the user last logged on to the computer.
`MACAddress`	Alpha-numeric text (maximum 256 characters). May be null.	The MAC address of the computer. This may be a comma-separated list if there is more than one active network adapter in the system. Do not include inactive network adapters and network adapters with invalid MAC addresses.
`MachineID`	Alpha-numeric text (maximum 100 characters). May be null.	For AIX, it is the System ID. For HP-UX, it is the Machine/Software ID. It is unset for other platforms.
`Manufacturer`	Alpha-numeric text (maximum 128 characters). May be null.	The manufacturer of the computer.
`MaxClockSpeed`	Unsigned integer (`int`). May be null.	The maximum clock speed of the fastest processor in the computer in kHz. Note that a number of server-based licenses depend on complete details of the processor types, counts and speeds to calculate a correct license position.
`MemoryUsage`	Unsigned integer (`bigint`). May be null.	The maximum memory usage of the virtual machine (bytes).
`ModelNo`	Alpha-numeric text (maximum 128 characters). May be null.	The model number of the computer.

Property	Attributes	Notes
`NumberOfCores`	Unsigned integer (`int`). May be null.	The total number of cores in the computer. If there is more than one physical processor in the computer, then this would be the sum of the core counts for all the processors. For example, in a computer with two quad-core processors, this value would be 8. Note that a number of server-based licenses depend on complete details of the processor types, counts and speeds to calculate a correct license position.
`NumberOfDisplay Adapters`	Unsigned integer (`int`). May be null.	The number of graphics cards in the computer.
`NumberOfHardDrives`	Unsigned integer (`int`). May be null.	The number of physical hard drives in the computer. While the intent is physical drives, often this can end up being the number of disk partitions.
`NumberOfLogical Processors`	Unsigned integer (`int`). May be null.	The number of logical processors in the computer. This is the number of 'execution contexts' the operating system has access to. It will commonly be equivalent to the number processors in a single core, non-multi-threaded processor architecture, to the number of cores in a multi-core single threaded processor architecture, and to the number of threads in a multi-threaded processor architecture. For example, in a two processor, quad-core and hyper-threaded computer, this value would be 16. Note that a number of server-based licenses depend on complete details of the processor types, counts and speeds to calculate a correct license position.
`NumberOfNetworkCards`	Unsigned integer (`int`). May be null.	The number of network cards in the computer.
`NumberOfProcessors`	Unsigned integer (`int`). May be null.	The total number of physical processors (CPU) in the computer. Note that a number of server-based licenses depend on complete details of the processor types, counts and speeds to calculate a correct license position.

Property	Attributes	Notes
`NumberOfSockets`	Unsigned integer (`int`). May be null.	The number of physical sockets into which a processor may be placed in the computer. It is rare that an inventory source can know this value. If unset, it is typically approximated by the number of processors.
`OperatingSystem`	Alpha-numeric text (maximum 128 characters). May be null.	The operating system of the computer. For virtual machines, it is the configured operating system of the guest. Note that this operating system identification is not used for licensing.
`PartialNumberOf Processors`	Fractional number (`float`). May be null.	Used in processor-based licensing, this is the non-integer number of cores allocated to this partition or virtual machine. When this property is null, the 'NumberOfCores' is used. Note that a number of server-based licenses depend on complete details of the processor types, counts and speeds to calculate a correct license position.
`PhoneNumber`	Alpha-numeric text (maximum 128 characters). May be null.	The phone number of the device. Used for mobile devices.
`PoolName`	Alpha-numeric text (maximum 100 characters). May be null.	The name of the pool that the virtual machine belongs to.
`PoolType`	Alpha-numeric text (maximum 100 characters). May be null.	The type of the pool that the virtual machine belongs to.
`ProcessorType`	Alpha-numeric text (maximum 256 characters). May be null.	The descriptive string of the processor(s) in the computer. This may be a comma-separated list in the case where there is more than one physical processor in the system. Note that a number of server-based licenses depend on complete details of the processor types, counts and speeds to calculate a correct license position.
`SerialNo`	Alpha-numeric text (maximum 100 characters). May be null.	The serial number of the computer.
`ServicePack`	Alpha-numeric text (maximum 128 characters). May be null.	The service pack installed for the operating system.

Property	Attributes	Notes
`TotalDiskSpace`	Unsigned integer (`bigint`). May be null.	The total size of all hard drives in the computer in bytes. Note that this can be a very large number on modern systems. The maximum value for a bigint is 9,223,372,036,854,775,807, which can represent about 9.2 exabyte. While in practice it is unlikely that this size of storage capacity is reached for a single system, some systems can end up with large values through virtualized drives. Therefore, it is worth considering capping values when calculating total disk space, particularly when converting values from kilobytes or megabytes to bytes.
`TotalMemory`	Unsigned integer (`bigint`). May be null.	The total RAM in the computer, in bytes.
`VMEnabledState`	Alpha-numeric text (maximum 128 characters). Default: 4. May be null.	The operational state of the virtual machine. If present, the value must be a (case insensitive) exact match to one of the values shown.
`VMLocation`	Alpha-numeric text (maximum 256 characters). May be null.	Location of the virtual machine on the file system.
`VirtualMachineType`	Alpha-numeric text (maximum 100 characters). May be null.	The type of the virtual machine. If present, the value must be a (case insensitive) exact match to one of the values shown.
`VirtualMachineUUID`	Alpha-numeric text (maximum 256 characters). May be null.	The unique identifier of the virtual machine provided by the virtualization infrastructure. (This may have the same value as the 'BIOSUUID', or have byte order reversed, or be altogether different.)

Inventory Object: ConsolidatedFileEvidence

`ConsolidatedFileEvidence` objects are uploaded to the `ConsolidatedFileEvidence` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

ConsolidatedFileEvidence provides a simpler interface to specify files and their usage on computers. It combines the computer, file evidence and usage details into a single row.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccessMode`	Alpha-numeric text (maximum 128 characters). Default: 1. Mandatory. Database key.	The access mode of the file evidence. Leave this blank unless this row is a virtualized application. In that case choose one of the values below that matches your application or desktop virtualization infrastructure. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`Company`	Alpha-numeric text (maximum 100 characters). Default: . Mandatory. Database key.	The company in the file header. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer. It must match the ComputerID from the Computer spreadsheet or the row will be ignored.
`Description`	Alpha-numeric text (maximum 200 characters). Default: . Mandatory. Database key.	The description in the file header. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`FileName`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The name of the file used as evidence of software installation. For unix operating systems include the full path in the file name, including the opening '/'. For Windows operating systems the file path is specified in the FilePath column and this column must only contain the file name.
`FilePath`	Alpha-numeric text (maximum 400 characters). May be null.	The path of the file used as evidence of software installation.
`FileSize`	Unsigned integer (`int`). Default: 0. Mandatory. Database key.	The size of the file in bytes. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`FileVersion`	Alpha-numeric text (maximum 100 characters). Default: . Mandatory. Database key.	The version number of the file used as evidence of software installation. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`Language`	Alpha-numeric text (maximum 200 characters). May be null.	The language in the file header.
`LastUsedDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The last used date of the usage.

Property	Attributes	Notes
`NumberOfSessions`	Unsigned integer (`bigint`). May be null.	The number of sessions that the file evidence was in use by the user specified in the UserID column during the usage tracking period. If multiple users used the same application on the computer, create one row for each user with usage.
`ProductName`	Alpha-numeric text (maximum 200 characters). May be null.	The product name in the file header.
`ProductVersion`	Alpha-numeric text (maximum 200 characters). May be null.	The product version number in the file header.
`StartDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The start date of the usage.
`UserID`	Unsigned integer (`bigint`). Mandatory. Database key.	The DOMAIN/SAMAccountName for the user that the file evidence was used by. If this software was used by multiple users, create one row for each user of the software on the computer. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: ConsolidatedInstallerEvidence

`ConsolidatedInstallerEvidence` objects are uploaded to the `ConsolidatedInstallerEvidence` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

ConsolidatedInstallerEvidence provides a simpler interface to specify installed applications and their usage on computers. It combines the computer, installer evidence and usage details into a single row.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccessMode`	Alpha-numeric text (maximum 128 characters). Default: 1. Mandatory. Database key.	The access mode of the installer evidence. Leave this blank unless this row is a virtualized application. In that case choose one of the values below that matches your application or desktop virtualization infrastructure. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer. It must match the ComputerID from the Computer spreadsheet or the row will be ignored.

Property	Attributes	Notes
`DatabaseName`	Unsigned integer (`bigint`). Mandatory. Database key.	If this installer evidence is an Oracle Database, then this field specifies the name of the database. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`DiscoveryDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The date that the installer evidence was first seen.
`DisplayName`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The display name of the software as reported by the installer evidence.
`Evidence`	Alpha-numeric text (maximum 32 characters). Default: . Mandatory. Database key.	Identifier for the type of installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`InstallDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The install date of the installer evidence.

Property	Attributes	Notes
`InstanceName`	Unsigned integer (`bigint`). Mandatory. Database key.	If this installer evidence is an Oracle Database, then this field specifies the name of the database instance. If there are multiple instances, create a row for each instance in this spreadsheet. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`LastUsedDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The last used date of the usage.
`NumberOfSessions`	Unsigned integer (`bigint`). May be null.	The number of sessions that the installer evidence was in use by the user specified in the UserID column during the usage tracking period. If multiple users used the same application on the computer, create one row for each user with usage.
`ProductCode`	Alpha-numeric text (maximum 55 characters). May be null.	The product code of the evidence. This is usually the MSI product code.
`Publisher`	Alpha-numeric text (maximum 200 characters). Default: . Mandatory. Database key.	The publisher of the software as reported by the installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`StartDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The start date of the usage.
`UserID`	Unsigned integer (`bigint`). Mandatory. Database key.	The DOMAIN/SAMAccountName for the user that the installer evidence was used by. If this software was used by multiple users, create one row for each user of the software on the computer. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`Version`	Alpha-numeric text (maximum 72 characters). Default: . Mandatory. Database key.	The version of the software as reported by the installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: ConsolidatedOracleDatabaseUser

`ConsolidatedOracleDatabaseUser` objects are uploaded to the `ConsolidatedOracleDatabaseUser` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

ConsolidatedOracleDatabaseUser provides a list of the users for each Oracle database instance.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccessMode`	Alpha-numeric text (maximum 128 characters). Default: 1. Mandatory. Database key.	The access mode of the installer evidence. Leave this blank unless this row is a virtualized application. In that case choose one of the values below that matches your application or desktop virtualization infrastructure. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`AccountStatus`	Alpha-numeric text (maximum 256 characters). May be null.	The current status of the end-user account.
`ComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer. It must match the ComputerID from the Computer spreadsheet or the row will be ignored.
`CreationDate`	Date/time field. May be null.	The date and time when the end-user was created.
`DatabaseName`	Unsigned integer (`bigint`). Mandatory. Database key.	This field specifies the name of the database. It must match a row in the InstallerEvidence spreadsheet for the same ComputerID or this row will be skipped.
`DefaultTablespace`	Alpha-numeric text (maximum 256 characters). May be null.	The default tablespace for an Oracle end-user.

Property	Attributes	Notes
`DisplayName`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The display name of the software as reported by the installer evidence. It must match a row in the InstallerEvidence spreadsheet for the same ComputerID, Version, Publisher, DatabaseName and InstanceName or this row will be skipped.
`Evidence`	Alpha-numeric text (maximum 32 characters). Default: . Mandatory. Database key.	Identifier for the type of installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`InstanceName`	Unsigned integer (`bigint`). Mandatory. Database key.	This field specifies the name of the database instance. If there are multiple instances, create a row for each instance in this spreadsheet. It must match a row in the InstallerEvidence spreadsheet for the same ComputerID and DatabaseName or this row will be skipped.
`LastLogonDate`	Date/time field. May be null.	The date and time when the end-user last logged on to the computer.
`Name`	Alpha-numeric text (maximum 256 characters).	The name of the user.
`Publisher`	Alpha-numeric text (maximum 200 characters). Default: . Mandatory. Database key.	The publisher of the software as reported by the installer evidence. It must match a row in the InstallerEvidence spreadsheet for the same ComputerID, DisplayName, Version, DatabaseName and InstanceName or this row will be skipped.
`TempTablespace`	Alpha-numeric text (maximum 256 characters). May be null.	The temporary tablespace for an Oracle end-user.
`UserID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the instance end-user. This may be an integer or a string.

Property	Attributes	Notes
`Version`	Alpha-numeric text (maximum 72 characters). Default: . Mandatory. Database key.	The version of the software as reported by the installer evidence. It must match a row in the InstallerEvidence spreadsheet for the same ComputerID, DisplayName, Publisher, DatabaseName and InstanceName or this row will be skipped.

Inventory Object: ConsolidatedRemoteAccessFile

`ConsolidatedRemoteAccessFile` objects are uploaded to the `ConsolidatedRemoteAccessFile` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The spreadsheet consolidates ImportedRemoteUserToApplicationAccess, ImportedFileEvidence and ImportedUser tables.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccessMode`	Alpha-numeric text (maximum 128 characters). Default: 1. Mandatory. Database key.	The AccessMode states how an application has been accessed. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`Company`	Alpha-numeric text (maximum 100 characters). Default: . Mandatory. Database key.	The company in the file header. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`Description`	Alpha-numeric text (maximum 200 characters). Default: . Mandatory. Database key.	The description in the file header. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`FileName`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The name of the file used as evidence of software installation. For unix operating systems include the full path in the file name, including the opening '/'. For Windows operating systems the file path is specified in the FilePath column and this column must only contain the file name.
`FilePath`	Alpha-numeric text (maximum 400 characters). May be null.	The path of the file used as evidence of software installation.

Property	Attributes	Notes
`FileSize`	Unsigned integer (`int`). Default: 0. Mandatory. Database key.	The size of the file in bytes. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`FileVersion`	Alpha-numeric text (maximum 100 characters). Default: . Mandatory. Database key.	The version number of the file used as evidence of software installation. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`Language`	Alpha-numeric text (maximum 200 characters). May be null.	The language in the file header.
`ProductName`	Alpha-numeric text (maximum 200 characters). May be null.	The product name in the file header.
`ProductVersion`	Alpha-numeric text (maximum 200 characters). May be null.	The product version number in the file header.
`ServerID`	Unsigned integer (`bigint`). Mandatory. Database key.	This is the ComputerID of the server that publishes this virtual application. The ComputerID must match a computer from the Computer spreadsheet, and that computer must have an installation of the application this file is part of. If the server does not have an installation of an appropriate application then the user will not be shown as having access to that application. This is a mandatory field.
`UserID`	Unsigned integer (`bigint`). Mandatory. Database key.	The fully qualified name of the user.

Inventory Object: ConsolidatedRemoteAccessInstaller

`ConsolidatedRemoteAccessInstaller` objects are uploaded to the `ConsolidatedRemoteAccessInstaller` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The spreadsheet consolidates ImportedRemoteUserToApplicationAccess, ImportedInstallerEvidence and ImportedUser tables.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccessMode`	Alpha-numeric text (maximum 128 characters). Default: 1. Mandatory. Database key.	The AccessMode states how an application has been accessed. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`DisplayName`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The display name of the software as reported by the installer evidence and is part of the unique identifier for installer evidence.
`Evidence`	Alpha-numeric text (maximum 32 characters). Default: . Mandatory. Database key.	The evidence type of the software as reported by the installer evidence and is part of the unique identifier for installer evidence.
`ProductCode`	Alpha-numeric text (maximum 55 characters). May be null.	The product code of the evidence. This is usually the MSI product code and is not part of the unique identifier.
`Publisher`	Alpha-numeric text (maximum 200 characters). Default: . Mandatory. Database key.	Publishers of software applications (for example, "Microsoft").
`UserID`	Unsigned integer (`bigint`). Mandatory. Database key.	The DOMAIN\SAMAccountName of the user.
`Version`	Alpha-numeric text (maximum 72 characters). Default: . Mandatory. Database key.	The version of the software as reported by the installer evidence and is part of the unique identifier for installer evidence.

Inventory Object: ConsolidatedVMPool

`ConsolidatedVMPool` objects are uploaded to the `ConsolidatedVMPool` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The VMPool spreadsheet provides a simple method to associate virtual machines with groups (pools) on their host.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`HostComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer which is hosting the pool. The HostComputerID should match the ComputerID in the Computer spreadsheet. Otherwise the record will be ignored.
`NumberOfCores`	Fractional number (`float`). May be null.	The number of cores in this pool.
`NumberOfProcessors`	Fractional number (`float`). May be null.	The number of processors in this pool.
`ObjectType`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The type of pool. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ParentName`	Alpha-numeric text (maximum 100 characters). May be null.	The name of the parent pool.
`ParentObjectType`	Alpha-numeric text (maximum 256 characters). May be null.	The type of pool of the parent.
`PoolFriendlyName`	Alpha-numeric text (maximum 256 characters).	The friendly name of the pool.
`PoolName`	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	The name of the pool.

Inventory Object:

ConsolidatedWMIEvidence

`ConsolidatedWMIEvidence` objects are uploaded to the `ConsolidatedWMIEvidence` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

ConsolidatedWMIEvidence provides a simpler interface to specify Windows Management Instrumentation (WMI) properties on computers. Other Web-Based Enterprise Management (WBEM) properties are supported from Unix computers as well. The most important data to provide in this spreadsheet is operating system installs. The 'Win32_OperatingSystem' class and the 'Name' property contains this data.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ClassName`	Alpha-numeric text (maximum 50 characters). Mandatory. Database key.	The WMI class name of the evidence. An example is 'Win32_OperatingSystem'.
`ComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer. It must match the ComputerID from the Computer spreadsheet or the row will be ignored.
`InstanceName`	Alpha-numeric text (maximum 256 characters). Default: . Mandatory. Database key.	The name of the WMI class instance. This is important when there a multiple instances of a WMI class on a computer. An example is the 'Win32_VideoController' class that may have many instances with the same properties. In this case you need to specify the name of the instance here, 'Intel(R) HD Graphics Family' or 'NVIDIA Quadro K2100M' for example. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`PropertyName`	Alpha-numeric text (maximum 50 characters). Mandatory. Database key.	The WMI property name of the WMI evidence. An example is 'Name'.
`PropertyValue`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The value of the property of the WMI evidence. An example is 'Microsoft Windows 7 Enterprise'

Inventory Object: Domain

`Domain` objects are uploaded to the `ImportedDomain` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedDomain` table holds all of the domains which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ComplianceDomainID`	Unsigned integer (`int`). May be null.	Identifier of the domain in the `ComplianceDomain` table that this imported domain links to. This is populated as part of the import process and does not need to be provided by the source connections.
`FlatName`	Alpha-numeric text (maximum 200 characters). Mandatory. Database key.	The flat name of the domain. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`QualifiedName`	Alpha-numeric text (maximum 200 characters). Mandatory. Database key.	The fully qualified name of the domain. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: EvidenceAttribute

`EvidenceAttribute` objects are uploaded to the `ImportedEvidenceAttribute` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedEvidenceAttribute` table holds all of the instance attributes from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AttributeID`	Unsigned integer (`int`). Mandatory. Database key.	The identifier used in the source connection for the instance attribute. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`AttributeName`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The name of the instance attribute. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: FileEvidence

`FileEvidence` objects are uploaded to the `ImportedFileEvidence` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedFileEvidence` table holds all of the file evidence which has been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccessModeID`	Unsigned integer (`int`). Default: 1. May be null.	The access mode ID of the file evidence.
`Company`	Alpha-numeric text (maximum 100 characters). May be null.	The company in the file header.
`Description`	Alpha-numeric text (maximum 200 characters). Default: ''.	The description in the file header.
`ExternalFileID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the file evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`FileName`	Alpha-numeric text (maximum 256 characters). May be null.	The name of the file used as evidence of software installation.
`FilePath`	Alpha-numeric text (maximum 400 characters). May be null.	The path of the file used as evidence of software installation.
`FileSize`	Unsigned integer (`int`). May be null.	The size of the file.
`FileVersion`	Alpha-numeric text (maximum 100 characters). May be null.	The version number of the file used as evidence of software installation.
`Language`	Alpha-numeric text (maximum 200 characters). May be null.	The language in the file header.
`ProductName`	Alpha-numeric text (maximum 200 characters). May be null.	The product name in the file header.

Property	Attributes	Notes
`ProductVersion`	Alpha-numeric text (maximum 200 characters). May be null.	The product version number in the file header.

Inventory Object: ILMTPVUCounts

`ILMTPVUCounts` objects are uploaded to the `ImportedILMTPVUCounts` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

This table allows the summarized PVU sub capacity numbers to be imported from ILMT. These numbers are calculated by ILMT for a particular date range as PVU "reports".

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ExternalNodeID`	Unsigned integer (`bigint`). Mandatory. Database key.	The external ID of the server to which these points apply.
`ExternalVMID`	Unsigned integer (`bigint`). Mandatory. Database key.	The external ID of the virtual machine associated with the node (server). ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`FullCapacityCores`	Unsigned integer (`int`). Default: 0.	The number of full-capacity licensable cores for the license on the computer.
`FullCapacityPVU`	Unsigned integer (`int`). Default: 0.	The number of full-capacity PVU counts consumed for the license on the computer.
`PeakFullCapacityPVU`	Unsigned integer (`int`). Default: 0.	The peak number of full-capacity PVU counts consumed for the license on the computer.
`PeakSubCapacityPVU`	Unsigned integer (`int`).	The peak number of sub-capacity PVU counts consumed for the license on the computer.
`Publisher`	An ASCII string of alphanumeric characters and punctuation (length 254 characters). Mandatory. Database key.	The name of the publisher of the title these points apply to.

Property	Attributes	Notes
`SubCapacityCores`	Unsigned integer (`int`). Default: 0.	The number of sub-capacity licensable cores for the license on the computer.
`SubCapacityPVU`	Unsigned integer (`int`). Default: 0.	The number of sub-capacity PVU counts consumed for the license on the computer.
`TitleName`	Alpha-numeric text (maximum 512 characters). Mandatory. Database key.	The name of the title these points apply to.

Inventory Object: InstalledFileEvidence

`InstalledFileEvidence` objects are uploaded to the `ImportedInstalledFileEvidence` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedInstalledFileEvidence` table holds a record of the file evidence that has been installed on a computer from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ExternalFileID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the file evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalFilePathID`	Unsigned integer (`bigint`). May be null.	The identifier used in the source connection for the path of the file evidence.

Property	Attributes	Notes
`ExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer that the file evidence is installed on. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: InstalledFileEvidenceUsage

`InstalledFileEvidenceUsage` objects are uploaded to the `ImportedInstalledFileEvidenceUsage` table in the operations (inventory) database.

The `ImportedInstalledFileEvidenceUsage` table holds a record of end-users that are using file evidence from the source connection.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ActiveTimeInSeconds`	Unsigned integer (`bigint`). May be null.	The number of seconds that the file evidence was in use during the usage tracking period.
`ExternalFileID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the file evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`ExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer that the file evidence is installed on. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalUserID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the end-user that has used the file evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`LastUsedDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The last used date of the file evidence.
`NumberOfSessions`	Unsigned integer (`bigint`). May be null.	The number of sessions that the file evidence was in use during the usage tracking period.
`StartDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The start date of the file evidence usage tracking period.

Inventory Object: InstalledInstallerEvidence

`InstalledInstallerEvidence` objects are uploaded to the `ImportedInstalledInstallerEvidence` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedInstalledInstallerEvidence` table holds a record of the installer evidence that has been installed on a computer from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`DiscoveryDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The date that the installer evidence was first seen.
`ExternalComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer that the installer evidence is installed on. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalInstaller EvidenceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`ExternalInstanceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the instance that the installer evidence is associated with. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`InstallDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The install date of the installer evidence.

Inventory Object: InstalledInstallerEvidenceAttribute

`InstalledInstallerEvidenceAttribute` objects are uploaded to the `ImportedInstalledInstallerEvidenceAttribute` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedInstalledInstallerEvidenceAttribute` table holds a record of the values of the instance attributes for each installer evidence which is reported to be installed on a computer.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AttributeID`	Unsigned integer (`int`). Mandatory. Database key.	The identifier used in the source connection for the instance attribute.

Property	Attributes	Notes
`ExternalComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer that the installer evidence is installed on. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalInstaller EvidenceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalInstanceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the instance that the installer evidence is associated with. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`Value`	Alpha-numeric text (maximum 2Gb storage, or about 1 billion double-byte characters).	The value of the instance attribute for the installed installer evidence.

Inventory Object: InstalledInstallerEvidenceUsage

`InstalledInstallerEvidenceUsage` objects are uploaded to the `ImportedInstalledInstallerEvidenceUsage` table in the operations (inventory) database.

The `ImportedInstalledInstallerEvidenceUsage` table holds a record of installed evidence being used from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer that the installer evidence is installed on. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalInstallerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`ExternalInstanceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the instance that the installer evidence is associated with. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalUserID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the user that the installer evidence was used on. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`LastUsedDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The last used date of the installed installer evidence.
`NumberOfSessions`	Unsigned integer (`bigint`). May be null.	The number of sessions that the installer evidence was in use during the usage tracking period.
`StartDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The start date of the installer evidence usage tracking period.

Inventory Object: InstalledWMIEvidence

`InstalledWMIEvidence` objects are uploaded to the `ImportedInstalledWMIEvidence` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedInstalledWMIEvidence` table holds a record of the WMI evidence that has been installed on a computer from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ExternalComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer that the WMI evidence is installed on. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalEvidenceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the WMI evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`InstanceName`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The name of the WMI class instance used in the source connection for the WMI evidence ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: InstallerEvidence

`InstallerEvidence` objects are uploaded to the `ImportedInstallerEvidence` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedInstallerEvidence` table holds all of the installer evidence which has been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccessModeID`	Unsigned integer (`int`). Default: 1. May be null.	The access mode ID of the file evidence.
`DisplayName`	Alpha-numeric text (maximum 256 characters). May be null.	The display name of the software as reported by the installer evidence.
`Evidence`	Alpha-numeric text (maximum 32 characters). May be null.	Identifier for the type of installer evidence.
`ExternalInstallerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`ProductCode`	Alpha-numeric text (maximum 55 characters). May be null.	The product code of the evidence. This is usually the MSI product code.
`Publisher`	Alpha-numeric text (maximum 200 characters). May be null.	The publisher of the software as reported by the installer evidence.
`Version`	Alpha-numeric text (maximum 72 characters). May be null.	The version of the software as reported by the installer evidence.

Inventory Object: InstallerEvidenceRepackageMapping

`InstallerEvidenceRepackageMapping` objects are uploaded to the `ImportedInstallerEvidenceRepackageMapping` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedInstallerEvidenceRepackageMapping` table is used by the importer to map the original and current installer evidence of repackaged softwares as reported by the ISO tag evidence.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`CurrentDisplayName`	Alpha-numeric text (maximum 256 characters). May be null.	The current display name of the repackaged software as reported by the ISO tag evidence.
`CurrentPublisher`	Alpha-numeric text (maximum 200 characters). May be null.	The current publisher of the repackaged software as reported by the ISO tag evidence.
`CurrentVersion`	Alpha-numeric text (maximum 72 characters). May be null.	The current version of the repackaged software as reported by the ISO tag evidence.
`OrigDisplayName`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The original display name of the repackaged software as reported by the ISO tag evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`OrigPublisher`	Alpha-numeric text (maximum 200 characters). Mandatory. Database key.	The original publisher of the repackaged software as reported by the ISO tag evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`OrigVersion`	Alpha-numeric text (maximum 72 characters). Mandatory. Database key.	The original version of the repackaged software as reported by the ISO tag evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: Instance

`Instance` objects are uploaded to the `ImportedInstance` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedInstance` table holds all of the instances which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AuditEvidence`	Binary data from a file (maximum 2Gb). May be null.	Oracle LMS CVS files in zip archive.
`AuditEvidenceDate`	Date/time field. May be null.	Oracle LMS CSV files collection date.

Property	Attributes	Notes
`ExternalComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`InstanceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the instance. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`InstanceName`	Alpha-numeric text (maximum 256 characters). May be null.	The name of the instance.
`ParentInstanceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the parent instance. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: InstanceUser

`InstanceUser` objects are uploaded to the `ImportedInstanceUser` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedInstanceUser` table holds all of the end-users of an instance which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccountStatus`	Alpha-numeric text (maximum 256 characters). May be null.	The current status of the end-user account.
`ApplicationID`	Alpha-numeric text (maximum 400 characters). Mandatory. Database key.	The Oracle EBS application ID the user has access to. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer.
`CreationDate`	Date/time field. May be null.	The date and time when the end-user was created.
`DefaultTablespace`	Alpha-numeric text (maximum 256 characters). May be null.	The default tablespace for an Oracle end-user.
`ExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the instance end-user.
`InstanceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the instance.
`LastLogonDate`	Date/time field. May be null.	The date and time when the end-user last logged on to the computer.
`TempTablespace`	Alpha-numeric text (maximum 256 characters). May be null.	The temporary tablespace for an Oracle end-user.

Inventory Object: LicenseUser

`LicenseUser` objects are uploaded to the `ImportedLicenseUser` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedLicenseUser` table holds all of the external end-users (such as those used by Oracle or SAP licenses) retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`CostCenter`	Alpha-numeric text (maximum 128 characters). May be null.	The cost center of the external end-user, as reported in SAP. Does not necessarily map to a cost centre in the `GroupEx` table.
`Description`	Alpha-numeric text (maximum 400 characters). May be null.	The description of the external end-user.
`Email`	Alpha-numeric text (maximum 400 characters). May be null.	An e-mail address for the external end-user.
`EmployeeNumber`	Alpha-numeric text (maximum 256 characters). May be null.	The employee number of the external end-user.
`ExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the external end-user. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`FirstName`	Alpha-numeric text (maximum 256 characters). May be null.	The first name of the external end-user.
`LastName`	Alpha-numeric text (maximum 256 characters). May be null.	The last name or surname of the external end-user.
`LicenseUserID`	Unsigned integer (`int`). May be null.	The identifier of the external end-user in the `LicenseUser` table that this imported end-user links to. This is populated by the import process and does not need to be provided by the source connections.

Property	Attributes	Notes
`UserName`	Alpha-numeric text (maximum 400 characters). May be null.	The name of the external end-user.

Inventory Object: RelatedInstalledInstallerEvidence

`RelatedInstalledInstallerEvidence` objects are uploaded to the `ImportedRelatedInstalledInstallerEvidence` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedRelatedInstalledInstallerEvidence` table holds parent-child relationship between installer evidence.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ChildExternal ComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer that the installer evidence is installed on. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ChildExternal InstallerEvidenceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`ConfidenceLevel`	Unsigned integer (`int`). May be null.	Confidence level for each bundled installer evidence (as a percentage).
`IsCharged`	Boolean (0 or 1). Mandatory. Database key.	The identifier used in the source connection to determine the pricing relation between parent and child installer evidence (specifies if it is charged = 1 or free = 0). ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ParentExternal ComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer that the installer evidence is installed on. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ParentExternal InstallerEvidenceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: RemoteUserToApplicationAccess

`RemoteUserToApplicationAccess` objects are uploaded to the `ImportedRemoteUserToApplicationAccess` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedRemoteUserToApplicationAccess` table stores the applications that remote users have access to

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccessModeID`	Unsigned integer (`int`). Mandatory. Database key.	The access mode ID for the remote application access. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalFileID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the file evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`ExternalInstallerEvidenceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalServerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The External Server ID for the remote server. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalUserID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the end-user that has used the file evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`LastUsedDate`	Date/time field. May be null.	The last time the remote application was used by the user.
`VDIGroupUUID`	A universally unique identifier. May be null.	The desktop group UUID from which the application is published

Inventory Object: Site

`Site` objects are uploaded to the `ImportedSite` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedSubnet` contains sites imported from Microsoft Active Directory

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AutoPopulated`	Boolean (0 or 1). Default: 0.	Is the site auto populated at source?
`Enabled`	Boolean (0 or 1). Default: 1.	Is the site enabled?
`Name`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The site's name.

Inventory Object: SiteSubnet

`SiteSubnet` objects are uploaded to the `ImportedSiteSubnet` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedSiteSubnet` contains sites and subnets imported from Microsoft Active Directory

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AutoPopulated`	Boolean (0 or 1). Default: 0.	Is the subnet auto populated at source?
`Enabled`	Boolean (0 or 1). Default: 1.	Is the subnet enabled?
`IPSubnet`	Alpha-numeric text (maximum 64 characters). Mandatory. Database key.	The IP subnet.
`IPSubnetBits`	UNRECOGNIZED TYPEMandatory. Database key.	The IP subnet mask in CIDR notation.
`SiteName`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The site's name.

Inventory Object: SoftwareLicense

`SoftwareLicense` objects are uploaded to the `ImportedSoftwareLicense` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedSoftwareLicense` table holds all of the licenses which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`EntitlementCount`	Unsigned integer (`int`). May be null.	The number of entitlements for the license.
`ExpiryDate`	Date/time field. May be null.	The expiry date of a subscription license.
`ExternalLicenseID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the license. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`IsSubscription`	Boolean (0 or 1). Default: 0.	Indicates whether or not the license is a subscription license.
`LicenseName`	Alpha-numeric text (maximum 256 characters). May be null.	The name of the license.
`PartNo`	Alpha-numeric text (maximum 100 characters). May be null.	The publisher's part number for this license.
`SoftwareLicenseID`	Unsigned integer (`int`). May be null.	Identifier of the license in the `SoftwareLicense` table that this imported license links to. This is populated by the import process and does not need to be provided by the source connections.
`SoftwareLicenseTypeID`	Unsigned integer (`int`). May be null.	The license type ID of the license.

Inventory Object:

SoftwareLicenseAllocation

`SoftwareLicenseAllocation` objects are uploaded to the `ImportedSoftwareLicenseAllocation` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedSoftwareLicenseAllocation` table holds the links between licenses and end-users which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ExternalLicenseID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the license. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalUserID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the license. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: User

`User` objects are uploaded to the `ImportedUser` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedUser` table holds all of the end-users which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ComplianceDomainID`	Unsigned integer (`int`). May be null.	Identifier of the domain in the `ComplianceDomain` table that this end-user belongs to. This is populated by the import process and does not need to be provided by the source connections.
`ComplianceUserID`	Unsigned integer (`int`). May be null.	Identifier of the end-user in the `ComplianceUser` table that this imported user links to. This is populated by the import process and does not need to be provided by the source connections.
`CostCenter`	Alpha-numeric text (maximum 128 characters). May be null.	The cost center of the end-user, as reported in SAP. Does not necessarily map to a cost centre in the `GroupEx` table.
`Domain`	Alpha-numeric text (maximum 100 characters). May be null.	The domain of the end-user.
`Email`	Alpha-numeric text (maximum 200 characters). May be null.	The email address of the end-user.
`EmployeeNumber`	Alpha-numeric text (maximum 128 characters). May be null.	The employee number of the end-user.
`ExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the end-user. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`FirstName`	Alpha-numeric text (maximum 128 characters). May be null.	The first name of the end-user.

Property	Attributes	Notes
InventoryAgent	Alpha-numeric text (maximum 64 characters). May be null.	The name of the person or tool that performed the last inventory. For imported spreadsheets, you may wish to include the name of the person preparing the data, in case there is subsequent follow-up required.
IsBlacklisted	Boolean (0 or 1). Default: 0.	This is populated by the import process and does not need to be provided by the source connections. The field is set to True if the end-user matches a record from the UserNameBlacklist table, meaning the account should not be included in compliance calculations.
LastName	Alpha-numeric text (maximum 128 characters). May be null.	The last name or surname of the end-user.
MapUsingEmailAddress	Boolean (0 or 1). Default: 0.	Indicates whether or not the user's email address should be used to try and map it to an existing ComplianceUser record.
SAMAccountName	Alpha-numeric text (maximum 64 characters). May be null.	The SAM account name of the end-user.
UserName	Alpha-numeric text (maximum 64 characters). May be null.	The account name of the end-user.

Inventory Object: VDI

VDI objects are uploaded to the ImportedVDI table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedVDIUser table stores the list of VDI devices, their master VM template and the VDI group the VDI device resides under.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ApplicationDelivery Only	Boolean (0 or 1). May be null.	Determines whether the VDI device is used only to server applications.

Property	Attributes	Notes
`BrokerType`	Alpha-numeric text (maximum 64 characters). Mandatory. Database key.	The broker type of the VDI device. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ComputerName`	Alpha-numeric text (maximum 64 characters). May be null.	The computer name of the VDI.
`Domain`	Alpha-numeric text (maximum 100 characters). May be null.	The domain name of the VDI device.
`ExternalDeviceID`	Unsigned integer (`bigint`). May be null.	The identifier used in the source connection for the VDI device.
`IsPersistent`	Boolean (0 or 1). May be null.	Determine whether the VDI device is a persistent VDI device.
`SiteName`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The site name of the VDI. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`TemplateName`	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	The VDI template the VDI is cloned from. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`VDIGroupName`	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	The VDI group the VDI device belongs to. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`VDIGroupUUID`	A universally unique identifier. May be null.	The group UUID the VDI device belongs to.

Inventory Object: VDITemplate

`VDITemplate` objects are uploaded to the `ImportedVDITemplate` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedVDITemplate` table stores the list of VDI templates.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`BrokerType`	Alpha-numeric text (maximum 64 characters). Mandatory. Database key.	The broker type of the VDI template. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`SiteName`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The site name of the VDI. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`TemplateName`	Alpha-numeric text (maximum 64 characters). Mandatory. Database key.	The template name of the VDI template. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`VDITemplateExternalID`	Unsigned integer (`bigint`). May be null.	The ExternalID of the VDI template in the `ImportedComputer` table.

Inventory Object: VDIUser

`VDIUser` objects are uploaded to the `ImportedVDIUser` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedVDIUser` table stores the list of users that have been granted access to VDI groups.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`BrokerType`	Alpha-numeric text (maximum 64 characters). May be null.	The broker type of the VDI for the end user.
`ExternalUserID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the end-user that has access to the VDI. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`SiteName`	Alpha-numeric text (maximum 256 characters). May be null.	The site name of the VDI.
`VDIGroupName`	Alpha-numeric text (maximum 100 characters). May be null.	The VDI group the end-user has access to.

Inventory Object: VMHostManagedBySoftware

`VMHostManagedBySoftware` objects are uploaded to the `ImportedVMHostManagedBySoftware` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedVMHostManagedBySoftware` table contains relationships between installer evidence of management software and VM hosts it manages.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ExternalComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer that the management software installer evidence is installed on.
`ExternalInstallerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for an installer evidence of management software.
`ExternalVMHostID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the VM host computer that is managed by a management software.
`RelationType`	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	Identifier for the type of relation, to be matched against ImporterString column of `RelationType` table.

Inventory Object: VMPool

`VMPool` objects are uploaded to the `ImportedVMPool` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedVMPool` table holds all of the virtual machine pools which have been retrieved from the source connections and the number of processors and cores that are assigned to each pool.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`HostComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer which is hosting the pool. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`NumberOfCores`	Fractional number (`float`). May be null.	The number of cores available to this pool.
`NumberOfProcessors`	Fractional number (`float`). May be null.	The number of processors available to this pool.

Property	Attributes	Notes
`ObjectType`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The type of pool. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ParentName`	Alpha-numeric text (maximum 100 characters). May be null.	The name of the parent pool. This is the PoolName property for the parent pool.
`ParentObjectType`	Alpha-numeric text (maximum 256 characters). May be null.	The type of pool of the parent.
`PoolFriendlyName`	Alpha-numeric text (maximum 256 characters). May be null.	The friendly name of the pool.
`PoolName`	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	The name of the pool. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: VirtualMachine

`VirtualMachine` objects are uploaded to the `ImportedVirtualMachine` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedVirtualMachine` table holds all of the virtual machines which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AffinityEnabled`	Boolean (0 or 1). Default: 0.	Set this to `True` if this VM is unable to move to different host computers.
`CPUAffinity`	Alpha-numeric text (maximum 256 characters). May be null.	Contains the CPU Affinity value for virtual machine(Host Logical Processors)
`CPUUsage`	Unsigned integer (`int`). May be null.	The maximum CPU usage of the virtual machine (MHz).
`ComputerName`	Alpha-numeric text (maximum 256 characters). May be null.	The computer name of the virtual machine.
`CoreAffinity`	Alpha-numeric text (maximum 256 characters). May be null.	Contains the Core Affinity value for virtual machine
`FriendlyName`	Alpha-numeric text (maximum 256 characters). May be null.	The friendly name of the virtual machine.
`GuestFullName`	Alpha-numeric text (maximum 256 characters). May be null.	Configured operating system for the guest.
`HostComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the virtual machine's host computer. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`InventoryAgent`	Alpha-numeric text (maximum 64 characters). May be null.	The name of the person or tool that performed the last inventory.
`Manufacturer`	Alpha-numeric text (maximum 128 characters). May be null.	The manufacturer of the virtual machine.
`MemoryUsage`	Unsigned integer (`bigint`). May be null.	The maximum memory usage of the virtual machine (bytes).
`ModelNo`	Alpha-numeric text (maximum 128 characters). May be null.	The model number of the virtual machine.
`NumberOfHardDrives`	Unsigned integer (`int`). May be null.	The number of hard drives in the virtual machine.
`NumberOfNetworkCards`	Unsigned integer (`int`). May be null.	The number of network cards in the virtual machine.

Property	Attributes	Notes
`NumberOfProcessors`	Unsigned integer (`int`). May be null.	The number of processors in the virtual machine.
`PartitionID`	Alpha-numeric text (maximum 100 characters). May be null.	Partition ID generated and used by the managing virtualization platform
`PartitionNumber`	Unsigned integer (`int`). May be null.	Number of this partition
`PoolName`	Alpha-numeric text (maximum 100 characters). May be null.	The name of the pool that the virtual machine belongs to.
`PoolType`	An ASCII string of alphanumeric characters and punctuation (length 100 characters). May be null.	The type of the pool that the virtual machine belongs to.
`ProcessorType`	Alpha-numeric text (maximum 256 characters). May be null.	The type of processor in the virtual machine.
`TotalMemory`	Unsigned integer (`bigint`). May be null.	The total RAM in the computer, in bytes.
`UUID`	Alpha-numeric text (maximum 256 characters). May be null.	The UUID of the virtual machine.
`VMComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the virtual machine's computer. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`VMEnabledStateID`	Unsigned integer (`int`). May be null.	The state of the machine (powered on, off, etc).
`VMLocation`	Alpha-numeric text (maximum 256 characters). May be null.	Location of the virtual machine on the file system.
`VMName`	Alpha-numeric text (maximum 256 characters). May be null.	The name of the virtual machine.

Property	Attributes	Notes
`VirtualMachineType`	An ASCII string of alphanumeric characters and punctuation (length 100 characters). May be null.	The type of virtual machine.

Inventory Object: WMIEvidence

`WMIEvidence` objects are uploaded to the `ImportedWMIEvidence` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedWMIEvidence` table holds all of the WMI evidence which has been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ClassName`	Alpha-numeric text (maximum 50 characters). May be null.	The WMI class name of the WMI evidence.
`ExternalEvidenceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the WMI evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`PropertyName`	Alpha-numeric text (maximum 50 characters). May be null.	The WMI property name of the WMI evidence.
`PropertyValue`	Alpha-numeric text (maximum 256 characters). May be null.	The value of the property of the WMI evidence.