FlexNet Manager Suite System Reference

Contents

Chapter 4: FlexNet Inventory Scanner..113

Chapter 5: Server Scheduling.. 143

Chapter 6: Inventory Adapter Studio... 176

Legal Information

Document Name: FlexNet Manager Suite 2015 R2 SP3 System Reference (for on premises implementations)

Part Number: FMS-11.3.0-SR01

Product Release Date: 17 February 2016

Copyright Notice

Intellectual Property

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

Restricted Rights Legend

Preface

System Reference

This document gathers together a range of reference material for FlexNet Manager Suite release 2015 R2 SP3. This forms part of a reference library that includes the chapters here, together with separate PDF files on larger topics such as adapters and the database schema. This grouping strikes a balance between supplying too many small PDF files, and a PDF file of unwieldy size.

1

Adding Custom Properties

Topics:

- *Custom Properties*

- *Objects You Can Customize*

- *Controls You Can Add*

- *Positioning Your Custom Control*

- *Creating a New Properties Tab*

- *Creating a New Section Within a Tab*

- *Creating Other Custom Properties*

- *Localizing Display Names of Custom Properties*

- *Removing a Custom Property*

- *Customizable Drop-Down Lists*

It is possible to add properties to underlying database objects, and have these custom properties displayed in the web interface. If you have an on premises implementation, with your own database, you can implement the changes yourself, following the guidelines in this chapter. If you use a cloud-based implementation, you can use these chapters to create a detailed specification of your requirements, and then submit a change request to your support contact from Flexera Software (or your third-party managed service provider) to implement your specified changes on your behalf.

Custom Properties

With a little technical effort, you can customize the properties of many objects presented in FlexNet Manager Suite.

The complexities of managing software licenses within your corporate processes inevitably means you will want additional fields to record data specific to your enterprise. This section explains how you can specify additional properties for various objects that are displayed in the property sheets and your custom reports within FlexNet Manager Suite.

Tip • *Since the 2015 R2 SP1 release of FlexNet Manager Suite, it is no longer necessary to download a SQL stored procedure to enhance the database for custom properties. This functionality is now built in to the database as installed.*

The broad overview of the process is:

1. Plan your custom property, including its control type, and where it should appear in the properties of its parent object.

2. In Microsoft SQL Server Management Studio, execute specific SQL procedures to declare your customization in a "top down" manner. For example, if you want an extra field in a new section of an entirely new tab of properties, you must first declare the tab, then the section, and then the field. Objects are positioned relative to others that already exist in the database. In running the procedures, you must also refer to all objects and properties by their internal names, or by numerical mappings. Those names and mappings are included below, in *Internal Property Names for Applications* on page 16 and the following similar topics.

3. Customizations are available immediately after the SQL procedure is run, so you can review the results immediate in the web interface for FlexNet Manager Suite. You can also immediately commence storing data in your new custom field through the web interface, as the database has been updated to store your data.

4. While the compliance database and the web interface are updated immediately, the data model for the Business Adapter Studio installed on your inventory beacon(s) is updated by a scheduled task running overnight. This means either waiting until next day before importing values with a business adapter to your new custom field; or you can trigger an immediate update as described in *Triggering Immediate Update of the BAS Data Model* on page 10.

Tip • *In general, customizations you make to the user interface and database are preserved through product upgrades. The one exception is the rare case where a product upgrade removes the 'anchor', the object used for positioning your custom control. In this case, both the anchor and your custom control disappear (although the data entered through the control is preserved and is still available in your customer reports). You can remedy this rare case by re-declaring the missing custom control relative to a new anchor. This restores your customization in the web interface, with full access to the previously-recorded data values.*

Limitations

In the web interface, a custom property is displayed in the property sheet of your chosen object, and it is automatically available in the report builder for inclusion in custom reports. However, the custom property is not available in the following:

- Standard, factory-supplied reports

- Grids in management views

- Search fields, including within property sheets.

As well, custom properties are always editable in property sheets (they cannot be made read-only in that context), and you cannot provide any validation to check data entered into the custom control.

In declaring internal names for your custom properties, you should adopt a stringent naming convention that starts with your own company name-space (a consistent abbreviation for your enterprise name, such as `AE` for Acme Enterprises). You'll next find it convenient to name the object type that you are adding (such as `Asset`). Finally, add the individual name of the property. Separate each of these naming elements with an underscore. Use only characters in the following ranges: `a-z`, `A-Z`, `0-9`, and underscore (_). (Specifically do not use a dot or dash.) A valid example name is:

```
AE_Asset_ChargeBackValue
```

Warning • *Do not use a naming convention that starts with the database object name and uses a dot as a separator. This combination produces obscure errors. Starting with your own name space makes it safe, and using an underscore separator also makes it safe.*

Triggering Immediate Update of the BAS Data Model

The data model exposed to the Business Adapter Studio installed on your inventory beacon(s) is updated by a scheduled task (`Regenerate Business Import config`) that runs overnight (by default, at 4am central server time). Therefore, if you add custom properties to FlexNet Manager Suite, you normally need to wait until next day before you can create a business adapter that loads data into your new custom field.

Alternatively, you can use the following process to trigger an immediate update to the data model for the Business Adapter Studio. This allows you to continue development from custom properties straight on to a custom business adapter that populates those properties.

1. On the batch server, open a Command Window.
2. Navigate to:

```
InstallDir\DotNet\bin\
```

The default value is

```
C:\Program Files (x86)\Flexera Software\FlexNet Manager Platform\DotNet\bin\
```

3. Execute the following command:

```
BatchProcessTaskConsole.exe run BusinessAdapterConfig
```

This launches the task that generates the updated data model for the Business Adapter Studio. To check when it is finished, run:

```
BatchProcessTaskConsole.exe list-tasks
```

While the task is running, `BusinessAdapterConfig` is visible in the task list, and it disappears within a few minutes, when the task is successful. The updated data model is then automatically collected by all inventory beacons when they "phone home" for updates. By default, this happens every 15 minutes, but the interval is configured in the web interface under **Discovery & Inventory** >**Settings**, under the **Beacon settings** section. Thereafter, restarting the Business Adapter Studio forces it to reload the data model.

4. After the propagation time, restart the Business Adapter Studio on your chosen inventory beacon. If you are running the Business Adapter Studio in connected mode on your central server, simply exit and restart. If you are running the Business Adapter Studio in disconnected mode on an inventory beacon, use this process::

   a) If the Business Adapter Studio is already open on your inventory beacon, you must exit and restart the entire FlexNet Beacon interface.

   b) In the FlexNet Beacon interface, navigate to the **Business Importer** page.

   c) Edit an existing import, or create a new one, and click **Edit adapter...**.
   Your newly-created custom properties are included in the list of available properties with a distinctive icon.

Objects You Can Customize

The following database objects support the addition of custom properties. When you are adding a custom property to any of these objects, you refer to them by the `TargetTypeID` listed here.

You can also make your custom property specific to only certain sub-types within each object (where available). For example, you may want to add a custom property to your assets, except that you don't want the custom property to appear on records of routers or switchers. You can exclude these two kinds of assets using the `TargetSubTypeID` included in the listing below. You reference the target sub-types using the numbers in the list (for example, refer to a workstation with the value `1`).

Table 1: Objects supporting customization

Object	TargetTypeID	TargetSubTypeID
Application	13	
Asset	9	1. Workstation 2. Server 3. Monitor 4. Desk

Object	TargetTypeID	TargetSubTypeID
		5. Chair
		6. Printer
		7. Router
		8. Switch
		9. Telephone
		10. Cellphone
		11. Laptop
		12. Mobile Device
Contract	10	1. General
		2. Lease
		3. Hardware Maintenance And Support
		4. Software License
		5. Software Maintenance And Support
		6. Blanket Purchase
		7. Consulting Services
		8. Insurance
		9. Rent
		10. Subscription
		11. Microsoft Business And Service Agreement
		12. Microsoft Select Agreement
		13. Microsoft Select Plus Agreement
		14. Microsoft Select Enrolment
		15. Microsoft Select Plus Enrolment
		16. Microsoft Enterprise Agreement
		17. Microsoft Enterprise Agreement Subscription
Purchases	20	1. Not Set
		2. Software
		3. Hardware
		4. Service

Object	TargetTypeID	TargetSubTypeID
		5. Other
		6. Software Upgrade
		7. Software Maintenance
		8. Disk Kit
		9. Hardware Maintenance
Software License	12	1. Enterprise
		2. Device
		3. Node Locked
		4. User
		5. Concurrent User
		6. Appliance
		7. Client Server
		8. OEM
		9. Evaluation
		10. Run Time
		11. Processor
		12. Site
		13. Named User
		14. Core
		15. Core Points
		16. Oracle DB Processor
		17. Oracle DB Named User Plus
		18. Processor Points
		19. Oracle DB Legacy
		20. Enterprise Agreement
		21. SAP Named User
		22. MS Server Processor
		23. CAL
		24. Tiered Device
		25. IBM PVU

Object	TargetTypeID	TargetSubTypeID
		26. IBM Authorized User
		27. IBM Concurrent User
		28. IBM Floating User
		29. Custom Metric
		30. One Point Per Processor
		31. IBM RVU
		32. IBM UVU
		33. MS Server Core
		34. Oracle Application User
		35. SAP Package
		36. MS SCCM Client Device
		37. MS SCCM Client User
		38. MSDN
Computer	14	**1.** Computer
		2. VM Host
		3. VM
		4. Remote Device
		5. Mobile Device
		6. VDI Template
User	15	

Controls You Can Add

When you declare a custom property to add to a database object, you must also declare the kind of control that is to appear in the property sheet for that object, within the web interface of FlexNet Manager Suite.

You can add an entirely new tab to the properties, or within any tab (new or existing) you can add a new section (a grouping for other controls). Both of these cases, tab and section, are special cases in that each has its own SQL procedure for its declaration. These do not need numeric references.

For other controls that you can add within a section (or, for that matter, within a tab without an intervening section if you wish), you specify them by a numeric reference called the `UIFieldTypeID`. The available controls and their `UIFieldTypeID` are shown below.

Prompts (the text beside the control telling the operator what to do) are declared as part of the declaration of each custom property.

Table 2: Supported UI controls

Control	UIFieldTypeID	Comments
Integer	3	A small single-line field combined with up and down arrows (a spinner), where operators may type in an integer value or use the arrows to 'spin up' the number required.
Text box	4	A standard, single-line text field for entering a value.
Text area	5	A rectangular area where the operator can enter free-form text.
Date	6	Provides a date entry field complete with date icon. If the operator clicks the icon, a date picker calendar appears.
Option list	8	A pull-down list of fixed values from which an operator may choose. You declare the values for the list when adding the custom property.
Check box	9	Boolean: saves a `1` in the database when the check box is checked (ticked), and zero otherwise.

Positioning Your Custom Control

Within the web interface for FlexNet Manager Suite, you declare the position of your newly-added custom property in relationship to something that already exists in the interface layout. This prior control may be one that came as standard in the factory-supplied interface, or may be another custom control that you have previously declared. We call this previously-existing control the 'anchor' for your newly-added custom property.

Your custom property can have various positional relationships with its anchor. These relationships are declared with numeric values.

Table 3: Positioning of custom controls relative to anchor (mandatory)

Positioning	UIInsertTypeID	Comments
Before	1	Before the anchor
After	2	After the anchor
Start of	3	At the start of an existing tab or section (not applicable for other types of anchor)

When you specify the anchor (the existing control from which your custom control is positioned), you do so by its name. If the anchor is another custom control that you declared earlier, you use exactly the name you specified then. If it is a factory-supplied control that is part of the standard web interface for FlexNet Manager Suite, you must use the internal database name for the anchor control. See the following sub-topics for the available objects, their controls as they appear in the English-language standard web interface (these values may be localized), and the internal and unchanging database name for the same control that you must reference as an anchor.

With the web interface, all properties pages support two columns of controls. When you insert a custom control, the columns reflow to accommodate the change, subject to the additional setting for each of the controls on the page.

For example, you can specify whether the custom control occupies just one column, or spans across two columns.

Table 4: Column span for custom controls

Columns	Width	Comments
Single column	1	Left or right column of the layout (default)
Double column	2	Always left aligned

For single-column controls, you may have a preference for whether the control appears on the left, or on the right, of the property page.

Table 5: Alignment of controls within page

Alignment	Position	Comments
Next available spot	0	The "don't care" option, where the control takes either left or right side based on its positioning in relation to the anchor (default).
Left column	1	This control is forced to the left. If its positioning is "After" an anchor that is already in the left column, the adjacent right side is left blank. Flow resumes after this control.
Right column	2	This control is forced into the right column. If its positioning is "After" an anchor that is already in the right column, the left side of the next line is left blank, and this control occupies the right.

Internal Property Names for Applications

The properties for applications are represented in the tables below, with a separate table for each tab displayed in the web interface (or UI). The first column (for sections with their own heading) and second column (for fields and other kinds of UI controls displayed within each section) show the labels displayed in the default culture eg-US. The right-most column displays the identity of that control within the underlying system. You must use these identity names when you reference any UI control as an anchor for relative positioning of your new custom control.

Limitations

Of all the database objects supporting custom values, only applications have the following limitations:

• The Business Importer does not current support importing data into custom fields for applications. When you create a custom property for applications, it is available for data entry within the web interface, and the data may be output in custom reports. This restriction is only that the bulk import of data using the Business Importer is not currently supported.

- The **Usage** tab is excluded from customization.

Tab label: General

Database identity: Tab_General

Section	Control	Internal Name
Identification		Section_Identification
	Table	Table
	Publisher	Publisher
	Version	Version
	Name	Name
	Source	Source
	FlexeraID	FlexeraID
	Classification	Classification
	Application category	Category
Details		Section_Details
	Status	Status
	Release date	ReleaseDate
	Supported until	SupportedUntil
	Extended support until	ExtendedSupportUntil
	Information	Notes

Tab label: Licenses

Database identity: Tab_Licenses

Section	Control	Internal Name
License consumption order		Section_LicenseConsumptionOrder
	Grid	LicenseOrder
	Automatically manage license priorities	ManagedLicenses

Tab label: Devices

Database identity: Tab_Computers

Section	Control	Internal Name
Related devices		Section_Computers

Section	Control	Internal Name
	Grid	RelatedComputers

Tab label: History

Database identity: Tab_History

Section	Control	Internal Name
History of changes to this application		Section_History
	Grid	ApplicationHistory

Internal Property Names for Assets

The properties for assets are represented in the tables below, with a separate table for each tab displayed in the web interface (or UI). The first column (for sections with their own heading) and second column (for fields and other kinds of UI controls displayed within each section) show the labels displayed in the default culture `eg-US`. The right-most column displays the identity of that control within the underlying system. You must use these identity names when you reference any UI control as an anchor for relative positioning of your new custom control.

Tab label: General

Database identity: Tab_General.

Section	Control	Internal Name
General		Section_General
	Name	Name
	Asset type	AssetTypeId
	Linked inventory	LinkedComputer
	Serial number	SerialNumber
	Asset tag	Asset_tag
	Manufacturer	Manufacturer
	Part number	PartNumber
	Model	AssetModel
	Master asset	MasterAssetName
	Status	Status
	Category	CategoryPath

Section	Control	Internal Name
End of life		Section_EndOfLife
	Retirement date	RetirementDate
	Resale price	ResalePrice
	Retirement reason	Reason
	Disposal date	DisposalDate
	Recipient	Recipient
	Written off value	WrittenOffValue
Last inventory		Section_LastInventory
	Electronic	Electronic
	By	Electronic_Created_By
	Physical	Physical
	By	Physical_Created_By
	Installed on	Installed
	Information	Note

Tab label: Hardware

Database identity: Tab_Hardware

Section	Control	Internal Name
Hardware		Section_Hardware
	Operating system	OperatingSystem
	Service pack	ServicePack
	Processors	Processors
	Cores	Cores
	Threads	Threads
	Sockets	Sockets
	Partial No Of Processors	PartialNoOfProcessors
	Processor type	ProcessorType
	Clock speed ñ MHz	ClockSpeed
	RAM GB	RAM
	Disk GB	Disk

Section	Control	Internal Name
	Hard drives	HardDrives
	Display adapters	DisplayAdapters
	Network cards	NetworkCards
	Assigned chassis type	AssignedChassisType
	Inventory Chassis Type	InventoryChassisType

Tab label: Ownership

Database identity: Tab_Ownership

Section	Control	Internal Name
User		*Do not reference.*
	Calculated	Calculated
	Last logged on	LastLoggedOn

Tab label: Financial

Database identity: Tab_Financial

Section	Control	Internal Name
Financial		Section_Financial
	Acquisition mode	AcquisitionMode
	Delivery date	DeliveryDate
	Warranty type	Warranty
	End of warranty	EndOfWarranty
Lease information *(see note)*		Section_LeaseInformation
	Lease agreement	LeaseAgreement
	Lease number	LeaseNumber
	Start date	StartDate
	Lease price	LeasePrice
	Buyout	BuyOut
	Payment	Payment
	Period	Period
Lease Termination *(see note)*		Section_LeaseTermination

Section	Control	Internal Name
	Date	TerminationDate
	Retirement reason	Reason
Depreciation		Section_Depreciation
	Current value	CurrentValue
	Residual value	ResidualValue
	Depreciation method	DepreciationMethod
	Period years	PeriodYears
	Rate	RatePercentage
Charges		Section_Charges
	Amount	Amount
	Frequency	Frequency

Note • *These sections and the fields they contain are applicable only when* **Application mode** *is set to* `Leased`.

Tab label: Sub-assets

Database identity: Tab_Sub_Assets

Section	Control	Internal Name
Sub-assets		Section_SubAssets
	Grid	SubAssets

Tab label: Contracts

Database identity: Tab_Contracts

Section	Control	Internal Name
Related contracts		Section_RelatedContracts
	Grid	AssociatedContracts

Tab label: Purchases

Database identity: Tab_Purchases

Section	Control	Internal Name
Related purchases		Section_RelatedPurchases
	Grid	AssociatedPurchases

Tab label: Documents

Database identity: Tab_Documents

Section	Control	Internal Name
Related documents		Section_Documents
	Grid	DocumentChanges

Tab label: History

Database identity: Tab_History

Section	Control	Internal Name
History of changes to this asset		Section_AssetsHistory
	Grid	History
	Created by	CreatedBy
	Creation date	CreationDate
	Last updated by	LastUpdatedBy
	Last updated date	LastUpdatedDate

Internal Property Names for Computers

The properties for computers are represented in the tables below, with a separate table for each tab displayed in the web interface (or UI). The first column (for sections with their own heading) and second column (for fields and other kinds of UI controls displayed within each section) show the labels displayed in the default culture `eg-US`. The right-most column displays the identity of that control within the underlying system. You must use these identity names when you reference any UI control as an anchor for relative positioning of your new custom control.

Tab label: General

Database identity: Tab_Summary.

Section	Control	Internal Name
General		Section_Summary

Section	Control	Internal Name
	Status	Status
	Inventory device type	InventoryDeviceType
	Name	Name
	Compliance status	ComplianceStatus
	Linked asset	LinkedToAsset
	Domain name	Domain
	Inventory role	Role
	Manufacturer	Manufacturer
	Model	ComputerModel
	IP address	IPAddress
	MAC address	MACAddress
	Serial number	SerialNumber
	Chassis number	ChassisNumber
	Category	CategoryPath
Last inventory source		Section_InventorySource
	Last inventory date	LastInventoryDate
	Last inventory source	LastInventorySource
	Connection name	InventoryConnectionName
	You may override inventory values	OverrideInventoryValue
Service Provider		Section_ServiceProvider
	Located in service provider's datacenter cloud	LocatedServiceProviderCloud
	Service Provider	ServiceProvider

Tab label: Hardware

Database identity: Tab_Hardware.

Section	Control	Internal Name
Hardware		Section_Hardware
	Operating system	OperatingSystem
	Service pack	ServicePack
	Processors	Processors

Section	Control	Internal Name
	Cores	Cores
	Threads	Threads
	Sockets	Sockets
	Partial No Of Processors	PartialNoOfProcessors
	Processor type	ProcessorType
	Clock speed ñ MHz	ClockSpeed
	RAM GB	RAM
	Disk GB	Disk
	Hard drives	HardDrives
	Display adapters	DisplayAdapters
	Network cards	NetworkCards
	Assigned chassis type	AssignedChassisType
	Inventory Chassis Type	InventoryChassisType
	* You may override inventory values	OverrideInventoryValue2

Tab label: Applications

Database identity: Tab_Applications.

Section	Control	Internal Name
Applications installed on this device		Section_Applications
	Grid	Applications

Tab label: Ownership

Database identity: Tab_Ownership.

Section	Control	Internal Name
User		*Do not reference.*
	Calculated	Calculated
	Last logged on	LastLoggedOn

Tab label: VM Properties

Database identity: Tab_VMProperties.

Section	Control	Internal Name
Virtual Machine Properties		Section_VMProperties
	Host	Host
	Name	VM_Name
	Guest full name	VM_GuestFullName
	UUID	VM_UUID
	Location	VM_Location
	VM type	VM_Type
	Pool	VM_Pool
	Total memory GB	VM_TotalMemory
	Memory usage GB	VM_MemoryUsage
	CPU usage MHz	VM_CPUUsage
	Last known state	VM_LastKnownState
	Affinity enabled	VM_AffinityEnabled

Tab label: Virtual Machines

Database identity: Tab_VirtualMachines.

Section	Control	Internal Name
Virtual machines		Section_VirtualMachines
	Grid	VirtualMachines

Tab label: Virtual Desktop Templates

Database identity: Tab_VdiTemplates.

Section	Control	Internal Name
Virtual Deskop Templates accessed from this device		Section_VdiTemplates
	Grid	VdiTemplates

Tab label: History

Database identity: Tab_History.

Section	Control	Internal Name
History of changes to this device		Section_History
	Grid	ComputerHistory
	Created by	CreatedBy
	Creation date	CreationDate
	Last updated by	LastUpdatedBy
	Last updated by	LastUpdatedDate

Internal Property Names for Contracts

The properties for software licenses are represented in the tables below, with a separate table for each tab displayed in the web interface (or UI). The first column (for sections with their own heading) and second column (for fields and other kinds of UI controls displayed within each section) show the labels displayed in the default culture eg-US. The right-most column displays the identity of that control within the underlying system. You must use these identity names when you reference any UI control as an anchor for relative positioning of your new custom control.

Tab label: General

Database identity: Tab_General.

Section	Control	Internal Name
Identification		Section_Identification
	Contract no	ContractNo
	Status	Status
	Description	Description
	Contract type	ContractType
	Purchase program	PurchaseProgram
	Select applications level	ApplicationLevel
	Select systems level	SystemsLevel
	Select servers level	ServersLevel
	Initial platform quantity	InitialPlatformQuantity
	Replaced by	ReplacedContractBy
	Category	CategoryPath

Section	Control	Internal Name
Events		Section_Events
	Evergreen	Evergreen
	Start date	StartDate
	Next renewal date	NextRenewalDate
	Expiry date	ExpiryDate
	Review date	ReviewDate
	Last reviewed on	LastReviewedOn
Payments		Section_Payments
	Global amount	GlobalAmount
	Monthly amount	MonthlyAmount
	Information	Comments

Tab label: Ownership

Database identity: Tab_ Ownership.

Section	Control	Internal Name
Ownership		Section_Ownership

Tab label: Vendors

Database identity: Tab_Vendors.

Section	Control	Internal Name
Other vendors		Section_OtherVendors
Additional vendors		Section_Vendors
	Grid	Vendors
Third-party vendors		Section_3rdPartyVendors
	Grid	Contract3rdPartyVendors

Tab label: Assets

Database identity: Tab_Assets.

Section	Control	Internal Name
Assets		Section_Assets

Section	Control	Internal Name
	Grid	Assets

Tab label: Purchases

Database identity: Tab_Purchases.

Section	Control	Internal Name
Related purchases		Section_RelatedPurchases
	Grid	Purchases

Tab label: Licenses

Database identity: Tab_Licenses.

Section	Control	Internal Name
Related licenses		Section_RelatedLicenses
	Grid	Licenses

Tab label: Responsibilities

Database identity: Tab_Responsibilities.

Section	Control	Internal Name
Responsibilities		Section_Responsibilities
	Grid	Responsibilities

Tab label: Payment schedules

Database identity: Tab_Payment_Schedule.

Section	Control	Internal Name
Payment schedules		Section_Payment_Schedule
	Grid	PaymentSchedules

Tab label: Terms and conditions

Database identity: Tab_Terms_and_Conditions.

Section	Control	Internal Name
Terms and conditions		Section_Terms_and_Conditions
	Grid	TermsConditions

Tab label: Documents

Database identity: Tab_Documents

Section	Control	Internal Name
Related Documents		Section_Documents Grid
	Grid	DocumentChanges

Tab label: History

Database identity: Tab_History.

Section	Control	Internal Name
History of changes to this contract		Section_History
	Grid	History
	Created by	CreatedBy
	Creation date	CreationDate
	Last updated by	LastUpdatedBy
	Last updated by	LastUpdatedDate

Internal Property Names for Licenses

The properties for software licenses are represented in the tables below, with a separate table for each tab displayed in the web interface (or UI). The first column (for sections with their own heading) and second column (for fields and other kinds of UI controls displayed within each section) show the labels displayed in the default culture `eg-US`. The right-most column displays the identity of that control within the underlying system. You must use these identity names when you reference any UI control as an anchor for relative positioning of your new custom control.

Tab label: Compliance

Database identity: Tab_Compliance.

Section	Control	Internal Name
Compliance		Section_Compliance
	Compliance status	ComplianceStatus
	Shortfall/Availability	Available
	Breach reason	BreachReason
Entitlements and consumption		EntitlementsAndConsumption

Section	Control	Internal Name
	Licensed from PO	PurchasedFromPO
	Allocated	NumberAllocated
	Extra entitlements	ExtraEntitlements
	Used	Used
	Raw consumption	RawConsumption
	Raw usage count	RawUserCount
	NUP minimum	NUPMinimum
	Peak consumed	PeakConsumption
	Raw installations	RawInstallation
	PUR savings	PURSavings
	Total licensed	TotalPurchased
	Consumed	AdjustedConsumption
	Consumption as at	ConsumedDate

Tab label: Identification

Database identity: Tab_Identification.

Section	Control	Internal Name
Identification		Section_Identification
	Publisher	Publisher
	Name	Name
	License type	LicenseType
	Subject to true up	SubjectToTrueUp
	Copy Version and Edition from the most recent application	CopyVersionEdition
	Version	Version
	Edition	Edition
	Duration	Duration
	Purchased as	PurchasedAs
	Expiry date	ExpiryDate
	Status	LifeCycle

Section	Control	Internal Name
	Retirement date	RetirementDate
	Retirement reason	RetirementReason
	Resale price	ResalePriceLink
	Metric	SoftwareLicenseMetricID
	Set Compliance status manually	ManuallySetComplianceStatus
	Resources consumed	ResourcesConsumed
	SAP type	SAPType
	Measurement date	MeasurementDate
	Tier type	TierType
	Tier code	TierCode
	Processors limit	ProcessorsLimit
	Cores limit	CoresLimit
	Legacy type	LegacyType
	Maximum sockets	MaximumSockets
	Minimum users	MinimumUsers
	Apply user limit per processor core	ApplyUserLimitPerCore
	Minimum processors	MinimumProcessors
	Points rule set	PointsRuleSet
	Category	CategoryPath
	Information	Notes
License keys		Section_LicenseKeys
	Rule	RuleType
	License key	LicenseKey

Tab label: Applications

Database identity: Tab_Applications.

Section	Control	Internal Name
Licensed software		Section_LicensedSoftware
	Title	ApplicationTitle
	Highest version	HighestVersion

Section	Control	Internal Name
Applications included in this license		Section_IncludedApplications
	Grid	Applications

Tab label: Purchases

Database identity: Tab_Purchases.

Section	Control	Internal Name
Purchase price		Section_PurchasePrice
	Override unit price	PurchasePrice
Purchases		Section_PurchaseOrderLineItems
	Grid	Purchases

Tab label: Financial

Database identity: Tab_Financial

Section	Control	Internal Name
Charges		Section_Charges
	Amount	Amount
	Frequency	Frequency
Resale		Section_Resale
	Resale price	ResalePrice
	Recipient	Recipient

Tab label: Contract

Database identity: Tab_Contracts.

Section	Control	Internal Name
Related contracts		Section_Contracts
	Grid	Contracts

Tab label: Consumption

Database identity: Tab_Consumption.

Section	Control	Internal Name
Normal user equivalents		Section_UserEquivalents
	Infrequent user	InfrequentUser
	External user	ExternalUser
Bulk user counts		Section_BulkUserCounts
	Additional infrequent users	AdditionalInfrequentUsers
	Additional external users	AdditionalExternalUsers
Related users		Section_RelatedEmployees
Related devices		Section_RelatedComputers
Alternate bulk user count		Section_AlternateBulkUserCount
	Non-inventoried users	AlternateNonInventoriedUsers
Users related to this license		Section_OracleUsers
	Grid	OracleUserConsumption

Tab label: Restrictions

Database identity: Tab_Restrictions.

Section	Control	Internal Name
Scope restrictions		Section_ScopeRestrictions
	Restrict to OS	Restrict_OS
	Grid	Restrictions

Tab label: Ownership

Database identity: Tab_Ownership.

Section	Control	Internal Name
Ownership and access rights		Section_Ownership

Tab label: Documents

Database identity: Tab_Documents

Section	Control	Internal Name
Related documents		Section_Documents
	Grid	DocumentChanges

Tab label: History

Database identity: Tab_History.

Section	Control	Internal Name
History of changes to this license		Section_History
	Grid	History
	Created by	CreatedBy
	Creation date	CreationDate
	Last updated by	LastUpdatedBy
	Last updated date	LastUpdatedDate

Internal Property Names for Purchases

The properties for purchases are represented in the tables below, with a separate table for each tab displayed in the web interface (or UI). The first column (for sections with their own heading) and second column (for fields and other kinds of UI controls displayed within each section) show the labels displayed in the default culture `eg-US`. The right-most column displays the identity of that control within the underlying system. You must use these identity names when you reference any UI control as an anchor for relative positioning of your new custom control.

Tab label: General

Database identity: Tab_General.

Section	Control	Internal Name
Purchase details		Section_PurchaseDetails
	Item	SequenceNumber
	Description	Description
	Part no./SKU	SKUDetails
	Purchase quantity	PurchaseQuantity
	Quantity per unit	QuantityPerUnit
	Effective quantity	EffectiveQuantity
	Publisher	Publisher
	Status	ItemStatus
	Purchase type	Type

Section	Control	Internal Name
Related contract		Section_RelatedContract
	Contract	Contract
Maintenance		Section_Maintenance
	Purchase includes support, maintenance, or other service Ö	MaintenanceOrServiceAgreemen
	Agreement date	AgreementDate
	Expiry date	ExpiryDate
Volume purchase program		Section_VolumePurchaseProgram
	Product pool	ProductPool
	Product points	ProductPoints
Notes		Section_Notes
	Comments	Comments

Tab label: Financial

Database identity: Tab_Financial

Section	Control	Internal Name
Invoice		Section_Invoice
	Invoice number	InvoiceNumber
	Invoice date	InvoiceDate
Shipping		Section_Shipping
	Shipping date	ShippingDate
	Shipping location	ShippingLocation

Tab label: Ownership

Database identity: Tab_Ownership.

Section	Control	Internal Name
Ownership		Section_Ownership

Assets

Database identity: Tab_Assets.

Section	Control	Internal Name
Assets		Section_Assets
	Grid	Assets

Tab label: Licenses

Database identity: Tab_Licenses.

Section	Control	Internal Name
Related licenses		Section_Licenses
	Allocate assigned entitlements	AllocateAssignedEntitlements
	Grid	Licenses

Tab label: Documents

Database identity: Tab_Documents.

Section	Control	Internal Name
Related documents		Section_Documents
	Grid	DocumentChanges

Tab label: History

Database identity: Tab_History.

Section	Control	Internal Name
History of changes to this purchase order		Section_History
	Grid	History
	Created by	CreatedBy
	Creation date	CreationDate
	Last updated by	LastUpdatedBy
	Last updated by	LastUpdatedDate

Internal Property Names for Users

The properties for users are represented in the tables below, with a separate table for each tab displayed in the web interface (or UI). The first column (for sections with their own heading) and second column (for fields and other kinds of UI controls displayed within each section) show the labels displayed in the default culture

`eg-US`. The right-most column displays the identity of that control within the underlying system. You must use these identity names when you reference any UI control as an anchor for relative positioning of your new custom control.

Tab label: General

Database identity: Tab_General.

Section	Control	Internal Name
Identification		User
	Title	UserTitle
	First name	FirstName
	Middle name	MiddleName
	Last name	LastName
	Suffix	Suffix
	Full name	Fullname
Employment		Section_Employment
	Job title	JobTitle
	Employee ID	EmployeeID
	Enployment Status	EmploymentStatus
	Manager	Manager
	Status	ComplianceUserStatus
Account		Section_Account
	Account name	AccountName
	Domain name	Domain
	Last inventory source	LastInventorySource

Tab label: Details

Database identity: Tab_Details.

Section	Control	Internal Name
Enterprise groups		Section_EnterpriseGroups
Contact		Section_Contact
	Phone	Phone
	Fax	Fax

Section	Control	Internal Name
	Mobile	Mobile
	Email	Email
Address		Section_Address
	Street address	StreetAddress
	City	City
	State/Province	State
	Country	Country
	Postal code	PostalCode

Tab label: Hardware

Database identity: Tab_Hardware.

Section	Control	Internal Name
Assets		Section_Assets
	Grid	Assets
Devices		Section_Computers
	Grid	Computers

Tab label: Software

Database identity: Tab_Software.

Section	Control	Internal Name
Related software licenses		Section_Software
	Grid	Software

Tab label: Responsibilities

Database identity: Tab_Responsibilities.

Section	Control	Internal Name
Responsibilities		Section_Responsibilities
Licenses		Section_Licenses
	Grid	Licenses
Contracts		Section_Contracts

Section	Control	Internal Name
	Grid	Contracts

Tab label: Documents

Database identity: Tab_Documents.

Section	Control	Internal Name
Related documents		Section_Documents
	Grid	DocumentChanges

Tab label: History

Database identity: Tab_History.

Section	Control	Internal Name
History of changes to this user		Section_History
	Grid	History
	Created by	CreatedBy
	Creation date	CreationDate
	Last updated by	LastUpdatedBy
	Last updated by	LastUpdatedDate

Internal Property Names for Vendors

The properties for vendors are represented in the tables below, with a separate table for each tab displayed in the web interface (or UI). The first column (for sections with their own heading) and second column (for fields and other kinds of UI controls displayed within each section) show the labels displayed in the default culture `eg-US`. The right-most column displays the identity of that control within the underlying system. You must use these identity names when you reference any UI control as an anchor for relative positioning of your new custom control.

Tab label: General

Database identity: Tab_General.

Section	Control	Internal Name
Identification		Section_Identification
	Name	Name
Contact information		Section_ContactInformation

Section	Control	Internal Name
	Phone	PhoneNumber
	Fax	Fax
	Email	Email
	Web	Web
Address		Section_Address
	Street address	StreetAddress
	City	City
	State/Province	State
	Country	Country
	Postal code	PostalCode

Tab label: Purchases

Database identity: Tab_Purchases.

Section	Control	Internal Name
Related purchases		Section_PurchaseOrderLineItems
	Grid	VendorPurchases

Tab label: Assets

Database identity: Tab_Assets.

Section	Control	Internal Name
Related assets		Section_AssociatedAssets
	Grid	VendorAssets

Tab label: Contracts

Database identity: Tab Contracts.

Section	Control	Internal Name
Related contracts		Section_AssociatedContracts
	Grid	VendorContracts

Tab label: History

Database identity: Tab_History.

Section	Control	Internal Name
History of changes to this vendor		Section_History
	Grid	History
	Created by	CreatedBy
	Creation date	CreationDate
	Last updated by	LastUpdatedBy
	Last updated by	LastUpdatedDate

Creating a New Properties Tab

Execute the following in SQL Server Management Studio against your FNMSCompliance database.

```
EXEC dbo.AddTabToWebUIPropertiesPage
      @TargetTypeID = TargetTypeID,
      @ExcludeTargetSubTypeIDs = 'TargetSubTypeID,TargetSubTypeID,...',
      @Name = 'My_Unique_Name',
      @CultureType = 'ISOCultureCode',
      @DisplayNameInPage = 'Prompt value',
      @UIInsertTypeID = UIInsertTypeID,
      @RelativeTabName = 'RelativeTabName'
```

where

@TargetTypeID	Mandatory. Integer that identifies the type of object to which you are adding a custom property. For supported objects and their integer equivalents for `TargetTypeID`, see *Objects You Can Customize* on page 11.
@ExcludeTargetSubTypeIDs	Mandatory. A comma-separated list (enclosed in single quotation marks) of integer subtype IDs. For the default case of no exclusions, give this parameter an empty list: `@ExcludeTargetSubTypeIDs = '',` Many types of target objects have subtypes (for example, assets may be workstations, routers, and so on). By default, a custom property added to an object (identified by its `TargetTypeID`) is added to all subtypes of that object. However, you can exclude any subtypes you choose with this parameter. For supported subtypes and their integer equivalents for *TargetSubTypeID*, see *Objects You Can Customize* on page 11.

@Name	Mandatory. The internal name (in code and database) of the new tab you are adding. This name must be unique across all tabs in the system (including the factory-supplied tabs, and including all database objects). For this reason, it is strongly recommended that you adopt a stringent naming convention, such as a company name space, an object type, and a tab name, each separated by an underscore (example: `MyCo_License_Chargebacks`). The name is limited to 256 characters. *Warning • Do not use a naming convention that starts with the database object name and uses a dot as a separator. This combination produces obscure errors. Starting with your own name space makes it safe, and using an underscore separator also makes it safe.*
@CultureType	Default value is `en-US`. Value is a five-character ISO code for culture (enclosed in single quotation marks). The permitted values are available at *http://msdn.microsoft.com/en-us/goglobal/bb896001.aspx*.
@DisplayNameInPage	Mandatory. This is the label (enclosed in single quotation marks) displayed on the tab in the web interface for FlexNet Manager Suite when the culture setting for the interface matched the one you declare in `CultureType`. You can also provide localized values for this label using different culture settings, for which see *Localizing Display Names of Custom Properties* on page 47.
@UIInsertTypeID	Mandatory. An integer indicating the position of this new tab relative to the tab identified in `RelativeTabName`. For integer values and their meaning, see *Positioning Your Custom Control* on page 15. Note that in this case of creating a new tab, the value 3 is not relevant.
@RelativeTabName	Mandatory. The internal name of the anchor tab relative to which you are positioning your new custom tab. For internal names of factory-supplied tabs, see subtopics of *Positioning Your Custom Control* on page 15.

Creating a New Section Within a Tab

Execute the following in SQL Server Management Studio against your FNMSCompliance database, referencing an existing tab.

```
EXEC dbo.AddSectionToWebUIPropertiesPage
        @TargetTypeID = TargetTypeID,
        @ExcludeTargetSubTypeIDs = 'TargetSubTypeID,TargetSubTypeID,...',
        @Name = 'My_Unique_Name',
        @CultureType = 'ISOCultureCode',
```

```
        @DisplayNameInPage = 'Prompt value',
        @TabName = 'tabInternalName',
        @UIInsertTypeID = UIInsertTypeID,
        @RelativePositionTo = 'RelativePositionTo'
```

where

@TargetTypeID	Mandatory. Integer that identifies the type of object to which you are adding a custom property. For supported objects and their integer equivalents for `TargetTypeID`, see *Objects You Can Customize* on page 11.
@ExcludeTargetSubTypeIDs	Mandatory. A comma-separated list (enclosed in single quotation marks) of integer subtype IDs. For the default case of no exclusions, give this parameter an empty list: `@ExcludeTargetSubTypeIDs = '',` Many types of target objects have subtypes (for example, assets may be workstations, routers, and so on). By default, a custom property added to an object (identified by its `TargetTypeID`) is added to all subtypes of that object. However, you can exclude any subtypes you choose with this parameter. For supported subtypes and their integer equivalents for *TargetSubTypeID*, see *Objects You Can Customize* on page 11.
@Name	Mandatory. The internal name (in code and database) of the new section you are adding. This name must be unique across all sections in the system (including the factory-supplied sections, across all database objects). For this reason, it is strongly recommended that you adopt a stringent naming convention, such as a company name space, an object type, optionally a tab name, and a section name, each separated by an underscore (example: `MyCo_License_Chargeback_General`). The name is limited to 256 characters. *Warning • Do not use a naming convention that starts with the database object name and uses a dot as a separator. This combination produces obscure errors. Starting with your own name space makes it safe, and using an underscore separator also makes it safe.*
@CultureType	Default value is `en-US`. Value is a five-character ISO code for culture (enclosed in single quotation marks). The permitted values are available at *http://msdn.microsoft.com/en-us/goglobal/bb896001.aspx*.

@DisplayNameInPage	Mandatory. This is the label (enclosed in single quotation marks) displayed above the section in the web interface for FlexNet Manager Suite when the culture setting for the interface matched the one you declare in `CultureType`. You can also provide localized values for this label using different culture settings, for which see *Localizing Display Names of Custom Properties* on page 47.
@TabName	Optional when `@UIInsertTypeID = 3`, and ignored for any other value. Provides the internal name the tab at the start of which the new section is to be inserted. If used, the tab name must be 80 characters or less, and enclosed inside single quotation marks. (If `TabName` is not specified, the value of `RelativePositionTo` is used.) For internal names of factory-supplied controls, see the subtopics under *Positioning Your Custom Control* on page 15.
@UIInsertTypeID	Mandatory. An integer indicating the position of this new section relative to the control identified in `RelativePositionTo`. For integer values and their meaning, see *Positioning Your Custom Control* on page 15. Note that in this case of creating a new section, the value `3` means at the start of the tab identified in `TabName`, meaning that `RelativePositionTo` is irrelevant in that case.
@RelativePositionTo	Mandatory when `UIInsertTypeID` has a value other than `3`; and when `@UIInsertTypeID = 3`, one of `RelativePositionTo` or `TabName` is required. The internal name of the anchor control relative to which you are positioning your new custom section. For internal names of factory-supplied controls, see the subtopics under *Positioning Your Custom Control* on page 15. When used with `@UIInsertTypeID = 3`, `RelativePositionTo` must be the name of a tab that has already been defined (and not any other kind of control).

Creating Other Custom Properties

Execute the following in SQL Server Management Studio against your FNMSCompliance database. The relative anchor from which positioning is determined must already be defined.

```
EXEC dbo.AddPropertyToWebUIPropertiesPage
     @TargetTypeID = TargetTypeID,
     @ExcludeTargetSubTypeIDs = 'TargetSubTypeID,TargetSubTypeID,...',
     @Name = 'My_Unique_Name',
     @CultureType = 'ISOCultureCode',
     @DisplayNameInPage = 'Prompt value',
     @DisplayNameInReport = 'Column heading',
     @TabName = 'MyTabName',
     @UIInsertTypeID = UIFieldTypeID,
     @UIFieldTypeID = UIFieldTypeID,
     @RelativePositionTo = 'RelativePositionTo'
     @SequenceNumber = 'IntegerCount'
```

```
        @Position = Position,
        @Width = Width,
        @DataSource = 'List, Of, Values',
        @DataSourceDelimiter = ',',
        @Required = 0,
        @StringLength = IntegerMaxLength,
        @ReadOnly = 0
```

where

@TargetTypeID	Mandatory. Integer that identifies the type of object to which you are adding a custom property. For supported objects and their integer equivalents for `TargetTypeID`, see *Objects You Can Customize* on page 11.
@ExcludeTargetSubTypeIDs	Mandatory. A comma-separated list (enclosed in single quotation marks) of integer subtype IDs. For the default case of no exclusions, give this parameter an empty list: ```@ExcludeTargetSubTypeIDs = '',``` Many types of target objects have subtypes (for example, assets may be workstations, routers, and so on). By default, a custom property added to an object (identified by its `TargetTypeID`) is added to all subtypes of that object. However, you can exclude any subtypes you choose with this parameter. For supported subtypes and their integer equivalents for *TargetSubTypeID*, see *Objects You Can Customize* on page 11.
@Name	Mandatory. The internal name (in code and database) of the new custom property you are adding. This name must be unique across all properties in the system (including the factory-supplied properties, across all database objects). For this reason, it is strongly recommended that you adopt a stringent naming convention, such as a company name space, an object type, and a property name, each separated by an underscore (example: `MyCo_License_DailyCharge`). The name is limited to 256 characters. *Warning • Do not use a naming convention that starts with the database object name and uses a dot as a separator. This combination produces obscure errors. Starting with your own name space makes it safe, and using an underscore separator also makes it safe.*
@CultureType	Default value is `en-US`. Value is a five-character ISO code for culture (enclosed in single quotation marks). The permitted values are available at *http://msdn.microsoft.com/en-us/goglobal/bb896001.aspx*.
@DisplayNameInPage	Mandatory. This is the label (enclosed in single quotation marks) displayed as a prompt in the web interface for FlexNet Manager Suite when the culture setting for the interface matched the one you declare in `CultureType`. You can also provide localized values for this label using different culture settings, for which see *Localizing Display Names of Custom Properties* on page 47.
@DisplayNameInReport	Mandatory. This is the label (enclosed in single quotation marks) displayed as a column heading in custom reports you prepare, when the culture setting for the interface matched the one you declare in `CultureType`. You can also provide localized values for this label using different culture settings, for which see *Localizing Display Names of Custom Properties* on page 47.

@TabName	Optional when `@UIInsertTypeID = 3`, and ignored for any other value. Provides the internal name the tab in which the new control is to be inserted. If used, the tab name must be 80 characters or less, and enclosed inside single quotation marks. (If `TabName` is not specified, the value of `RelativePositionTo` is used.) For internal names of factory-supplied tabs, see the subtopics under *Positioning Your Custom Control* on page 15. For details about creating custom tabs, see *Creating a New Properties Tab* on page 41.
@UIInsertTypeID	Mandatory. An integer indicating the position of this new section relative to the control identified in `RelativePositionTo`. For integer values and their meaning, see *Positioning Your Custom Control* on page 15. Note that in this case of creating a new section, the value `3` means at the start of the tab identified in `TabName`, meaning that `RelativePositionTo` is irrelevant in that case.
@UIFieldTypeID	Mandatory. An integer indicating the kind of control used to display your custom property. For integer values and their meaning, see *Controls You Can Add* on page 14.
@RelativePositionTo	Mandatory. The internal name of the anchor control relative to which you are positioning your new custom section. For internal names of factory-supplied controls, see the subtopics under *Positioning Your Custom Control* on page 15.
@SequenceNumber	Optional (when omitted, the default value is null). Where two or more custom properties are declared with the same anchor in their @RelativePositionTo parameters, they are ordered by the sequence number. If there is no sequence number declared, they are ordered by the execution order of the SQL commands.
@Position	Optional (when omitted, the default value is `0`). The alignment of your custom control within the two-column layout of a properties page. For the integer values and their meanings, see *Positioning Your Custom Control* on page 15.
@Width	Optional (when omitted, the default value is `1`). The number of columns spanned by this control in the two-column layout of a properties page. For more information, see *Positioning Your Custom Control* on page 15.
@DataSource	Mandatory when `UIFieldTypeID = 8`, and otherwise ignored. Within single quotes, this is an ordered list of the values to be displayed within the option list. By default, the list is comma-separated, but see @DataSourceDelimiter. Values (between delimiters) may include white space, and leading white space on a value is ignored. Every value must be unique. One of the values may be a null, creating a blank row in the option list in the web interface. Example: `@DataSource = ', Apples, Oranges, Ripe Pears, Tangerines'` This example creates an option list with the first position blank (this displays an empty value until the operator selects another value from the list). ☞ *Restriction • When you add a custom option list (when `UIFieldTypeID = 8`), it is not possible to localize the values for the individual options within the custom option list. (This is in contrast to adding a custom option within a drop-down list included in the standard product: the standard lists allow for customization of any*

@DataSourceDelimiter	Optional (when omitted, the default value is the comma ,). A single ASCII character (a punctuation character is expected) that does not occur in your data set and is used as a delimiter between values in @DataSource. The separator character must be enclosed in single quotation marks. If `UIFieldTypeID` has any value other than `8`, this parameter is ignored.
@Required	Optional (when omitted, the default value is zero). May have the following values: • `0` means that input to the custom field in the web interface is optional, such that the value may be left blank. • `1` means that in the web interface, the custom field is mandatory, and may not be left blank by an operator completing the enclosing tab. This integer value must *not* be surrounded by quotation marks. *Note • This parameter affects only data input through the web interface of FlexNet Manager Suite. It does not have any effect, for example, on data imports using the Business Importer.*
@StringLength	Optional (when omitted, the default value is `256`). Ignored unless the @UIFieldTypeID has either of the values `4` (text box, or field) or `5` (text area, multi-line). Specifies the maximum length of the input string in the web interface. The largest permissible string length is 4000 bytes.
@ReadOnly	Optional (when omitted, the default value is zero). May have the following values: • `0` means that the control is read/write, and can be updated in the web interface. • `1` means that the control is read-only, and cannot be changed in the web interface. This value is illegal if `@Required = 1`, and will produce an error when executed.

(Row above @DataSourceDelimiter continues: *options, including added custom options; whereas option lists that are in entirety custom cannot be localized.)*)

Localizing Display Names of Custom Properties

Execute the following in SQL Server Management Studio against your FNMSCompliance database, once the custom properties already exist in the database.

```
EXEC dbo.CustomPropertyUpdateDisplayName
        @Name = 'My-Unique-Name',
        @CultureType = 'ISOCultureCode',
        @DisplayNameInPage = 'Prompt value',
```

```
        @DisplayNameInReport = 'Column header'
```

where

@Name	Mandatory. The internal name (in code and database) of your custom property, declared when you added it to the database. Never localize this internal name.
@CultureType	Mandatory. A five-character ISO culture name (enclosed in single quotation marks). The permitted values are available at *http://msdn.microsoft.com/ en-us/goglobal/bb896001.aspx*. This is the culture for the localized values in this declaration. Notice that localized values are only displayed in FlexNet Manager Suite when your enterprise has installed the corresponding language pack that provides localized values for the factory-supplied controls as well.
@DisplayNameInPage	Mandatory. This is the localized label (enclosed in single quotation marks) displayed on the tab in the web interface for FlexNet Manager Suite when the culture setting for the interface matched the one you declare in `CultureType`.
@DisplayNameInReport	Mandatory. The localized label (enclosed in single quotation marks) that is available for you to include in custom reports that display this custom property.

You can repeat this procedure as often as required to define localized display names in all cultures in use in your enterprise.

Tip • The appearance of the web interface for different locales is controlled in two separate places:

• *The formatting of dates and numeric values is taken from the settings on your web browser.*

• *The language presented in the web interface is controlled in the* **My Preferences** *page (under the configuration menu available in the top-right corner of the web interface. Options for other languages are only present when your enterprise has purchased appropriate language pack options.*

This separation allows for the common case where a single language (such as English) may have different date formats (such as 12/31/14 and 31/12/14) in different parts of the world.

Removing a Custom Property

Defined a custom property incorrectly? Execute one of the following in SQL Server Management Studio against your FNMSCompliance database, referencing the faulty custom property.

Warning • Removing a custom property (tab, section, or other control) in one of the following ways deletes the control from the web interface of FlexNet Manager Suite, removes the custom property from the custom report

builder, and optionally can also delete the property from the underlying database. If you specify removal of the data from the database (`@DeleteFromDB = 1`), any custom reports previously built that included the custom property will fail to load until you modify the report definition to remove this custom property.

Deletion is specific to each individual custom property you have previously declared, and does not cascade down to other items they may contain. For example, suppose you had declared `My.License`**`Tab`**`.Charges`, containing `My.License`**`Section`**`.Monthly`, in which there was a custom control `My.Chargeback.Amount`. Subsequently you delete `My.LicenseTab.Charges`. Because the tab disappears, obviously everything it contained is also 'hidden'. However, `My.LicenseSection.Monthly` and `My.Chargeback.Amount` have not been deleted, and are waiting in the database. You can repeat the creation process for these controls using the same name for each (this performs an update), and declaring a new anchor point for them in the web interface (for instance, in this example you might move them into the **Financial** tab). Thereafter these controls reappear in the web interface, and display any data previously saved through the custom controls.

```
EXEC dbo.RemoveTabFromWebUI
      @TargetTypeID = TargetTypeID,
      @Name = 'My-Unique-Name'
```

```
EXEC dbo.RemoveSectionFromWebUI
      @TargetTypeID = TargetTypeID,
      @Name = 'My-Unique-Name'
```

```
EXEC dbo.RemovePropertyFromWebUI
      @TargetTypeID = TargetTypeID,
      @Name = 'My-Unique-Name',
      @DeleteFromDB = 0
```

where

@TargetTypeID	Mandatory. Integer that identifies the type of object from which you are removing the custom property. For supported objects and their integer equivalents for `TargetTypeID`, see *Objects You Can Customize* on page 11.
@Name	Mandatory. The internal name (in code and database) of the custom property you are removing. You defined this name when you create the custom property.
@DeleteFromDB	Optional (default is zero when omitted). Boolean. When omitted or given the value zero, the custom property (and the data it may contain if it has already been in use) remains in the database, although it is no longer available in the web interface of FlexNet Manager Suite. When this parameter is set to 1, the custom property is removed (along with any stored values) from the database. This parameter is only available when removing properties, as tabs and sections do not have any data component stored in the database.

Customizable Drop-Down Lists

Operators can select values from drop-down lists of available options. If you have system administration rights, you can customize some of these lists. The following table contains all customizable drop-down lists and their corresponding SQL database tables.

`AcquisitionMode`	A list of methods that can be used to obtain an asset. You may add methods, but it is recommended that you do not rename or remove the existing items. Examples include `Purchased`, `Leased`, `Rented`.
`AssetStatus`	The status of an asset. You can add states, but existing items must not be renamed or removed. Examples include `Purchased`, `In storage`, `Retired`. Assets with a status of `Retired` or `Disposed` do not consume from software licenses during compliance calculations. No other status values (including any you may add) affect consumption calculations.
`AssetWarrantyType`	The type of warranty that applies to an asset. This list can be replaced with your own values. Examples include `One year on site`, `Three year on site`.
`ComplianceComputerRole`	The kinds of role that a computer might have. This list can be replaced with your own values. Examples include `Failover`, `Training`, `Backup`, `Test`.
`ComplianceUserStatus`	The status of a user within an organization. You can add status values, but existing items must not be renamed or removed. Examples include `Active`, `Retired`, `Pending` (perhaps for an employee just starting with the company).[1]
`ComputerChassisType`	The chassis type, that is, that casing, of a computer. You can add types, but existing items must not be renamed or removed. Examples include `Notebook`, `Sealed-Case PC`, `Mini Tower`.
`ContractType`	The type of a contract. You can add types, but existing items must not be renamed or removed. Examples include `Lease`, `Support`.
`ContractStatus`	The state of a contract. You can add states, but existing items must not be renamed or removed. Examples include `Active`, `Completed`, `Draft`.
`EmploymentStatus`	A user's type of employment within your organization. This list can be replaced with your own values. Examples include `Employee`, `Part-time`, `Casual`.
`EndOfLifeReason`	A list of disposal methods for an asset. This list can be replaced with your own values. Examples include `Lost`, `Stolen`, `Sold`.
`InstanceRole`	The possible roles of an instance of a piece of software (such as an Oracle database). In some cases, vendors can attribute different licensing terms to an instance depending on its role within the organization. This list can be replaced with your own values. Examples include `Backup`, `Failover`, `Mirroring`.

---

[1] The `ComplianceUserStatus` list differs from the other lists in that it has an additional `IsUserActive` property that must be specified whenever a new status entry is added. Set the value of this property to `1` if you want end users with your new status to be included in compliance calculations, and `0` if you want them omitted. By default, end users with a status of `Retired` or `Inactive` are not counted in license reconciliation calculations, because they are assumed to not be active members of the organization.

InstanceEnvironment	The possible environments in which an instance of software can be deployed. For some vendors, the environment may affect the cost of licensing. This list can be replaced with your own values. Examples include `Development`, `Staging`, `Production`.
LeaseEndReason	A list of reasons why a lease can be terminated. This list can be replaced with your own values. Examples include `Early Termination - Asset Returned`, `Buyout`, `Trade`.
LicenseStatus	A list of license states. You can add new status values, but existing items must not be renamed or removed. Examples include `Active`, `Retired`.
OracleLegacyLicenseType	While the compliance of your older Oracle licenses is not tracked automatically, you may still record them and manually set their compliance status. Update this list to include any Oracle legacy license types in use at your organization. It is recommended that you do not rename or remove existing items. Examples include `Concurrent Device`, `Developer Network License`.
PurchaseOrderDetailType	The possible types of purchases. You can add types to this list, but existing items must not be renamed or removed. Examples include `Software`, `Software Upgrade`, `Hardware maintenance`.
ResponsibilityType	A list of responsibilities that can be assigned to users who are involved in contract management. You can add types to this list, but it is recommended that you do not rename or remove existing items. Examples include `Owner`, `Signatory`, and `Interested Party`.
SoftwareLicenseDuration	A list of license periods. You can add items to this list, but it is recommended that you do not rename or remove existing items. Examples include `Perpetual`, `Time Limited`.
SoftwareTitleClassification	A classification for the applications used in your enterprise. You can add items to this list, but existing entries must not be renamed or removed. Examples include `Freeware`, `Malware`, `Update`.
TermAndConditionType	Shown as the **Type** of a term or condition in the properties of a contract, it is used to classify the terms or conditions being documented for a contract. You can customize these types only on the **Terms and conditions** tab of the contract properties. Examples include `Acceptance Period`, `Renewal`, `Limitation`.
UserTitle	The titles that can be used before a person's name. This list can be extended with your own values. Examples include `Mr.`, `Miss`.
UserSuffix	Additional information about the user's name, for example, `Jr.`, `Sr.`, and so on. This list can be extended with your own values.

Columns in SQL DB tables

The table above lists the SQL database tables that you can modify to add, change, or remove entries. Each of these tables contains least three columns:

• An identifier field (for example, `UserTitleID`).

Remember •   If you add a new value, it is automatically assigned a unique identifier greater than 1000. The first 1000 identifiers are reserved for internal use, do not manually adjust this field.

- A resource name column (usually called `ResourceString`). This is a unique field that must exist for every row. It is used to find the internationalized text to be displayed on the screen so that different language versions of FlexNet Manager Platform will display different text for the same value. It should contain the name of a resource string to be loaded from the internationalized system. If the string does not exist for a particular language version, then the value in the default value column will be used instead. Every value in the `ResourceString` column is also represented as a row in the `ComplianceResourceString` table.

Tip •   Because options in a drop-down list change infrequently, they are cached to improve database performance. Therefore, after adding a custom option in one of the drop-down lists mentioned in the table above, restart Microsoft IIS on the web application server (or, on smaller implementations, the server hosting that functionality).

- The default value (usually called `DefaultValue`). It is displayed on the user interface if no localized string is available.

Customizing a drop-down list

Tip •   You might need to check if the drop-down list you are about to modify can be customized: for details, see Customizable Drop-Down Lists on page 50.

Restriction •   While you can use this process to add a custom option (including localization) within a standard option list shipped with FlexNet Manager Suite, it is not currently possible to localize the options within an entirely new custom option list.

Before adding, removing, or changing a value on one of the customizable lists, you must make an equivalent change to the `ComplianceResourceString` database table. You can then update the database table that corresponds to the drop-down list you are modifying.

1. Run SQL Server Management Studio and modify the rows in the `ComplianceResourceString` database table.
2. Open an SQL database table that corresponds to the drop-down list you want to customize, and add the new values.

Note •   The values that you add must be unique within the table they are being added to.

The following SQL example illustrates how two entries (Dame and Earl) are added to the ComplianceResourceString and UserTitle tables, and how the ResourceStringCultureType table is updated:

```
INSERT INTO dbo.ComplianceResourceString (ResourceString)
VALUES ('UserTitle.Dame')
INSERT INTO dbo.ComplianceResourceString (ResourceString)
VALUES ('UserTitle.Earl')
INSERT INTO dbo.UserTitle (DefaultString, ResourceString)
VALUES ('Dame', 'UserTitle.Dame')
INSERT INTO dbo.UserTitle (DefaultString, ResourceString)
VALUES ('Earl', 'UserTitle.Earl')
INSERT INTO dbo.ResourceStringCultureType (ResourceString, CultureType,
 ResourceValue)
VALUES ('UserTitle.Dame', 'en-US', 'Dame')
INSERT INTO dbo.ResourceStringCultureType (ResourceString, CultureType,
 ResourceValue)
VALUES ('UserTitle.Earl', 'en-US', 'Earl')
--- Refer to the ComplianceCultureType table for a list of valid language
--- values for your enterprise.
```

2

Importing Inventory Spreadsheets and CSV Files

Topics:

- *Overview of Inventory Spreadsheets*

- *One-Off Import of an Inventory Spreadsheet*

- *Setting Up Scheduled Imports of Inventory from Spreadsheets*

- *Viewing Validation Errors for Uploaded Inventory Spreadsheets*

- *Deleting Spreadsheet Inventory Data from the Database*

Among many supported methods of importing software and hardware inventory details, FlexNet Manager Suite supports the import of inventory information in either comma-separated value (`.csv`) files, or Microsoft Excel spreadsheets (`.xlsx` files). To differentiate these from other imports of spreadsheet information (such as for purchases, enterprise groups, and user assignments), these are called 'inventory spreadsheets', a convenient term covering both file formats (unless specifically excepted).

Inventory spreadsheets can be imported in either of two ways:

- A 'one-off' import through the web interface of FlexNet Manager Suite

- A repeatable, scheduled import through an inventory beacon.

This chapter covers the processes for both kinds of imports of inventory spreadsheets.

The formats of these imported files are fixed, and defined by downloadable templates. The documentation of each of these templates, and the mapping of all the spreadsheet columns to the compliance database, is included in the *Inventory Spreadsheet Templates* chapter of the companion volume, *FlexNet Manager Suite Schema Reference*.

Overview of Inventory Spreadsheets

You can upload inventory data from spreadsheet files to FlexNet Manager Suite in two ways:

- A one-off, one-time upload that you might need to do for workflow validation, demonstration or proof-of-concept purposes.

  The data will be saved on the application server and created as a unique connection. Once data from this connection is processed, the connection is disabled (and cannot be re-enabled). Inventory from this connection ages, and eventually appears in the **Out-Of-Date Inventory** list. To clear the stale data, delete the connection: everything imported (only) from that connection is then removed from the operations databases.

- Ongoing import of spreadsheet inventory data.

  To set up this workflow, you must use an inventory beacon. This workflow allows scheduled, repeated imports, and data updates, in just the same way as regular inventory imports from other (non-spreadsheet) data sources.

  ☞

  Remember • The complete set of inventory data should be uploaded whenever updating the data previously imported from a pre-existing spreadsheet. That is, all original rows, along with the new rows of inventory data, get imported every time the data has to be updated. As with all other inventory imports, records that disappear from the source connection (in this case, your spreadsheets) are automatically removed from the operations databases to maintain synchronization with the data source.

If you need to collect inventory from a source FlexNet Manager Suite cannot directly connect to (for example, a source inaccessible for security or any other reasons), you can export this software, hardware or virtualization data from your source into a comma-separated value (`.csv`) or Excel (`.xlsx`) file. This inventory spreadsheet acts as an intermediate file for data upload through an inventory beacon. For repeated use, you can use custom, in-house scripts to populate these spreadsheets with current data, saving them to your chosen upload folder. You can then schedule regular imports from your upload folder to keep the data current.

The following inventory types are supported:

- Computers and VMs

- Installation evidence

- File evidence

- Windows Management Instrumentation (WMI) evidence

- Oracle evidence

- Virtual machine (VM) pool data

- Cluster evidence

- Cluster group data

- Cluster host affinity rule data

- Remote access file and install evidence (use these templates for lists of all users who can access a cloud-based service, or virtual desktops and applications).

Each type of inventory has a fixed file format, and you can access the corresponding inventory templates in either of two ways:

- You can download them from the web interface of FlexNet Manager Suite

- You can open them from an inventory beacon, where local copies are automatically synchronized with the central application server.

You can then populate your chosen templates with your inventory data, and import the completed files back into FlexNet Manager Suite.

One-Off Import of an Inventory Spreadsheet

You can manually upload your inventory data from a spreadsheet to FlexNet Manager Suite.

The **Inventory Data One-Off Upload** page (accessible through the **Inventory Data** tab of the **Data Inputs** page) is for a single import of inventory data from spreadsheets. If you need to repeat the inventory data import (for example to make a data correction), you must first delete the previously set-up connection (the corresponding server-side copy of the spreadsheet is automatically deleted as well). For details, see *Deleting Spreadsheet Inventory Data from the Database* on page 62.

Each one-off upload creates a separate import connection. This is true even if two uploads have an identical name: updates are not possible through the web interface, and two uploads of the same name produce two identically-named connections, functioning separately.

🔲

Important • Do not allow two or more one-time connections for inventory spreadsheets to exist at the same time, even with different import names. Data from every upload is persisted on the central application server, and is imported afresh from the central spreadsheet copies into the operations databases for every inventory import and license calculation. Having multiple connections to spreadsheets that contain the same computers (for example, from the mandatory `Computer` *spreadsheet, reflected in lists of inventory devices) can cause data 'toggling' between imported values, based on the which connection for spreadsheet data was most recently processed. Therefore you must delete the previous one-off upload connection before uploading a newer batch of inventory data.*

Scheduling regular imports of inventory spreadsheets is not supported through the web interface: it is a one-time connection. Once data from this connection is processed, the connection is disabled (and cannot be re-enabled). Inventory from this connection ages, and eventually appears in the **Out-Of-Date Inventory** list. To clear the stale data, delete the connection: everything imported (only) from that connection is then removed from the operations databases. (In contrast, you can arrange regularly scheduled spreadsheet imports through an inventory beacon, as described in *Setting Up Scheduled Imports of Inventory from Spreadsheets* on page 59.)

The data from an individual spreadsheet file may affect several database tables in FlexNet Manager Suite. For more details about the template's column names and their corresponding database fields, see the corresponding section in *FlexNet Manager Suite Schema Reference*. You can download this document from the title page of the online help.

☞

Remember • *Within your spreadsheets, the column names and column order cannot be modified from those supplied in the template files. Any such change results in an import failure. (Here, for a one-off import, you may rename the spreadsheet files themselves, since their purpose is made clear by the field through which you identify and upload each spreadsheet. In contrast, when the same templates are used on an inventory beacon for scheduled uploads, the file names as well as the column names and column order must all be maintained.)*

1. Navigate to the system menu (⚙▼ in the top right corner), and click **Data Inputs**.

2. Click the **Inventory Data** tab, and ensure that the inventory data **Name** which is marked as `Primary` has a **Task status** of `Completed`.

   At least the first occurrence of the primary inventory import must have completed successfully before any secondary source (including the one-off inventory spreadsheet) can be imported. If you attempt a one-off import of an inventory spreadsheet before the first successful primary import, it results in an error `Inventory import failed because data has not been imported from the primary data source.` This is because of the rules for data merging, explained in more detail in *Making a Data Source Connection the Primary One* on page 60. By default, the primary connection is the internal inventory connection, named **FlexNet Manager Suite** in this list. If it has not yet completed, a member of the `Administrator` role can run an import and compliance calculation manually by navigating to **License Compliance** > **Reconcile** (in the **Events** group). Be sure to select **Update inventory for reconciliation** (available only to administrators) before clicking **Reconcile**.

3. Click **One-off upload**.

4. Download the `InventoryTemplates-version.zip` file, and populate the template that you need with the inventory data.

   💡

   Tip • *When you process spreadsheets uploaded through the **Application virtualization** section, there are two possible paths:*

   - *You may want to record consumption against existing users on their computers that are already recorded in the operations databases. In this case, be certain that the user's ID from the central database is exactly recorded in the `UserID` column in (either or both of) the spreadsheets used for **Application virtualization**, which are identified in the **Access shown by file evidence** or **Access shown by installer evidence** fields of this **Inventory Data One-Off Upload** page. When the user is matched, the installation is recorded against a computer that the user 'owns' (that is, is linked as either the assigned user or calculated user).*

   - *You may want to create new records for remote devices and remote users who are not already recorded in your operations databases. To do this, make sure that both these statements are true:*

     - *The `Computer` spreadsheet (identified in the **Computers and VMs** field) contains data in both the `ComputerName` and `LastLoggedOnUser` columns.*

     - *The value in that `LastLoggedOnUser` column matches the value in the `UserID` column in (either or both of) the spreadsheets used for **Application virtualization**, which are identified in the **Access shown by file evidence** or **Access shown by installer evidence** fields.*

5. In the **Upload name** field, enter a name for this inventory data upload.

It is best practice to specify an easily recognizable name, because this name is used in lists of data connections. In particular, when the time comes to delete the connection to this one-off import, you will value a self-evident name. Perhaps consider a name space prefix, such as `OOIIS-` or some other convention to help you isolate one-off imports of inventory spreadsheets.

6. From the **Spreadsheet type** option list, select the inventory file format you are going to upload.

*Attention • The files are processed depending on the option you selected from the **Spreadsheet type** list. Therefore in a single upload, you can include several distinct inventory files (one of each of the kinds listed on this page), but within a single upload all of the uploaded files must be of the same spreadsheet type.*

7. For each kind of inventory that you wish to import from spreadsheets:

   a) Next to the field for the appropriate data type, click **Browse**, and select the matching `.csv` or `.xlsx` template-based file containing your inventory data.

   ☞

   *Restriction • As a minimum, you must upload one file through the **Computers and VMs** field. This file is mandatory because it contains computer names and serial numbers, plus the `Processor Cores` field required for license optimization.*

   b) Next to each identified data file, click **Upload**.
   "`File uploaded successfully`" message is displayed.

   c) Repeat the identification and upload process for all files included in this named upload.

8. Scroll down to the bottom of the web page, and click **Start processing**.
   The name of your inventory connection is added to the table at the bottom of the page, and `In progress` gets displayed in the **Status** column. When the inventory file is fully processed, and the license reconciliation is finalized, `Completed` is displayed instead. Note that all inventory sources area reconciled, regardless of type.

Depending on the file type you imported, its data is available from the corresponding area of the web interface. For example, if you imported computer-related data, navigate to **Discovery & Inventory** >**All Inventory** to view the uploaded records.

*Note • License reconciliation does not imply that there are no validation errors: you might need to click [+] in the **Task**/**Step** name column to see the results of individual steps. If there are any errors, click the hyperlink and troubleshoot as needed. For example, an error that occurred during the `Import into staging` step indicates an issue with the staging, in-memory tables or an invalid spreadsheet type. File-import tasks with the `Failed` status are displayed below the ⚙▼ menu, and are indicated with a red dot:*

You can also view a detailed log of any step within the inventory import: click [+] to expand its task, and in the **Logs** column for the step in question, click **Download log**.

Setting Up Scheduled Imports of Inventory from Spreadsheets

On an inventory beacon, you can set up a repeatable, automatic uploading of selected spreadsheet files of inventory data to FlexNet Manager Suite.

Important • If it is not the first time that you are importing a spreadsheet file into FlexNet Manager Suite, the values already imported with the previous file will be updated and overwritten accordingly. Any update of a previously uploaded spreadsheet file:

- *Updates the data saved from any modified rows.*

- *Inserts data found in new rows.*

- *Deletes the data from the operational database that was previously imported from rows that have since been removed from the new spreadsheet file.*

- *Deletes duplicate rows. If a duplicate row is identified, only a single entry is created. Identification is based on matching the values in key columns. For example, if the keys match, but some of the other data is different, the first row of the two is kept, while all duplicate rows that follow are discarded.*

Templates are available through the inventory beacon (as described below), or can reuse templates downloaded for one-off uploads through the web interface of FlexNet Manager Suite (provided that these have not been renamed).

1. Start FlexNet Beacon.

Note • To run FlexNet Beacon, you must have system administrator rights.

2. in the **Data collection** group, click **Inventory systems**.
3. Click the arrow on the **New** button, and click **Spreadsheet**.
4. From the **Spreadsheet Type** option list, select the type to be used.
5. To prepare your inventory spreadsheets:

   a) Click the **Templates** hyperlink displayed in the **Create Spreadsheet Source Connection** dialog box.

   b) Populate the template(s) you choose with the appropriate type of inventory data.

   When scheduling an automatic, scheduled inventory import, you can populate and update spreadsheet files either by a purpose-built script, or through a work flow applicable to your organization. It is good

practice to edit spreadsheet files in a work folder separate from your upload folder. This prevents a clash between file updates and file uploads that could result in incomplete data being processed.

Important • Remember that you cannot change the file name, nor any columns (number, names, or order) supplied in the template.

c) Create an upload folder, and save your completed inventory spreadsheet files to that folder.

All spreadsheet files located in the one folder are included in the scheduled uploads.

*Note • The folder can also be located on a shared network drive (make sure that an account running the inventory beacon has **Read** permissions on the folder).*

6. In the **Connection Name** field, type in a name for this inventory data upload.

7. In the **Connection Folder** field, browse to the folder with inventory spreadsheets you created in the steps above.

*Tip • If you do not want to import the inventory data as yet, select the **Connection is in test mode (do not import results)** check box. (Remember that data from any secondary inventory source, including this one, cannot be imported until after the first successful import from your primary inventory source.)*

8. Select the overlapping inventory filter that you need.

9. Click **Save**.

The new connection is added to the list of available inventory connections.

10. Select the new connection, and either:

   • Click **Execute Now**.

   • Click **Schedule...**, and choose the schedule on which to run repeated imports from the **Connection Folder**. (For details on creating schedules, see the online help for the inventory beacon.)

Making a Data Source Connection the Primary One

If you import hardware inventory fields from multiple sources, some fields duplicated across the sources may receive conflicting data. A common case is that one inventory tool returns a particular hardware property, while another tool does not collect the same property and returns a null. There are also differences in the ways tools collect inventory, so that sometimes values vary across tools.

Tip • This section applies only to hardware inventory values. Software inventory is always merged across all sources regardless of any source being marked as primary.

FlexNet Manager Suite resolves conflicting hardware data in two ways:

- A non-null value received from an inventory connection designated as primary is never overwritten. (Nulls can be replaced.)

- Among values received from secondary sources, generally data with the most recent inventory date is used.

Note • There are some other settings for the secondary sources (related to whether duplicate inventory should be merged, ignored, or ignored only if older than x days).

For example, suppose you have three inventory sources that report these values for a single device A:

Source	Primary	Last inventory date	Cores	Threads
Source 1		10 May 2015	4	16
Source 2	Yes	12 May 2015	8	NULL
Source 3		14 May 2015	6	NULL

All non-null hardware properties from Source 2 are given priority, because it is the primary source. Thereafter, based on date, Source 3 is used, and finally Source 1. The final record for Device A shows 8 cores and a total of 16 threads.

An inventory spreadsheet is treated exactly like any other inventory connection in this regard. You can use a repeated import of an inventory spreadsheet to correct specific values reported incorrectly by other inventory tools.

Tip • You cannot make a one-off inventory spreadsheet upload primary. After a single upload, this source is disabled, and its inventory data ages. Only a repeated upload of an inventory spreadsheet through an inventory beacon should be considered as a possible primary source. Even then, you cannot make it primary until after its first import, so that the source is recognized by the central application server.

To make a source connection the primary one, click **Make primary** displayed on its row on the **Inventory Data** tab of the **Data Inputs** page of the web interface.

Viewing Validation Errors for Uploaded Inventory Spreadsheets

Diagnose the source of any spreadsheet validation errors.

A page is available that analyzes validation errors in all uploaded inventory spreadsheets (both one-off through the web interface, and scheduled through an inventory beacon). This page is not directly available through the menus, but you can reach it in either of the following ways:

1. To access through the **Inventory Data One-Off Upload** page:

   a) Scroll to the bottom to the **Last 5 uploads** list.

   b) If necessary, click the **+** expander in the **Task**/**Step** column to reveal individual steps in the processing until the error is revealed.

   c) In the **Summary** column, click the **Validation errors** hyperlink.
      The **Inventory Upload Validation Errors** page is displayed.

2. To access through the **Data Inputs** page:

   a) Navigate to the system menu (⚙ ▼ in the top right corner), and click **Data Inputs**.

   b) Click the **Inventory Data** tab.

   c) Identify the connection for your inventory spreadsheet import, and click the expander arrow on its far right. The **Last completed import** section shows the count of validation errors. You may click the count (if it is more than zero).

   d) If necessary, click on **Show**/**hide task status and history** to expose the matching panel.

   e) In the **Summary** column, click the **Validation errors** hyperlink.
      The **Inventory Upload Validation Errors** page is displayed.

3. Use the diagnostic information to locate and fix the problem. (See the online help for this page for further details.)

4. Repeat the upload using the modified spreadsheet(s).

   Important • For a one-off upload, remember that you must delete the connection for your previous upload before attempting another.

Deleting Spreadsheet Inventory Data from the Database

To remove data imported from an inventory spreadsheet, delete its connection.

When any inventory data source is removed from FlexNet Manager Suite, the data imported exclusively from that source is removed from the database as well.

Tip • Any data imported from multiple sources remains until its last source is removed. This means that if you want to delete from the database those inventory records that you imported only from a spreadsheet, you need only remove the connection to that inventory spreadsheet.

You may want to delete the connection to an inventory spreadsheet for several possible reasons:

- You made a mistake with some values in a one-time import of an inventory spreadsheet. To correct this, you must first delete the previous connection to that spreadsheet, and then do a new one-off upload of the amended inventory spreadsheet.

- A one-time upload failed in some way, and is now disabled. You must delete this connection to retry.

- You accidentally have multiple connections to one-time inventory spreadsheet imports existing to exist at the same time. All but one of these must be deleted.

- Inventory imported from a one-time spreadsheet import has aged, and you want to remove it from the **Out-Of-Date Inventory** listing.

- You want to change details about a scheduled import of inventory spreadsheets through an inventory beacon.

You can delete the connections separately for:

- One-time data uploads (for details about one-off uploads, see *One-Off Import of an Inventory Spreadsheet* on page 56). These connections are deleted only in the web interface.

- Scheduled, repeated uploads through an inventory beacon (for more information about these uploads, see *Setting Up Scheduled Imports of Inventory from Spreadsheets* on page 59). Connections established on an inventory beacon must be deleted both in the web interface and separately in the FlexNet Beacon interface.

Starting in the web interface for FlexNet Manager Suite, use this process to delete a connection to an inventory spreadsheet.

1. Navigate to the system menu (⚙ ▼ in the top right corner), and click **Data Inputs**.

2. Click the **Inventory Data** tab.

   💡

   Tip • For connections through inventory beacons, if you do not want to delete your set-up connection, and plan to re-use it in future, select **Disabled** *from the* **Connection status** *option list displayed on its row. Manually disabled connections can be re-enabled when you are ready. (In contrast, one-off upload connections are automatically disabled after a single processing run, and cannot be re-enabled.)*

3. Click the light-grey triangle displayed at the far right of your data connection's row:

   A panel expands to reveal more details about the connection.

4. Inside this panel, click **Delete connection**.

5. Click **OK** on the confirmation dialog.

6. If this is a scheduled inventory spreadsheet import (that is, one running through an inventory beacon):

   a) Log in to the appropriate inventory beacon (for example, through Remote Desktop Connection), and start FlexNet Beacon.

      💡

      Tip • You must log in with an account that has administrator privileges on the inventory beacon.

   a) Ensure that the **Inventory systems** page is selected, and select the connection from the list.

Your connection shows `Spreadsheet` in the **Type** column.

b) Click **Delete**, and on the confirmation dialog, click **OK**.

The saved copy of your spreadsheet is removed (from the central application server for one-off uploads, or from the inventory beacon for repeatable uploads). At the next import and compliance calculation, the records created from that spreadsheet are removed from the database.

3

Oracle Discovery and Inventory

Topics:

Discovery and inventory information is a prerequisite for performing license compliance calculations in FlexNet Manager Suite. In addition to the general inventory collection features, FlexNet Manager Suite also offers specialized features for Oracle inventory collection. With a detailed description of each of the supported methods, this chapter may help you in deciding and implementing the appropriate inventory collection method for Oracle systems in your computing estate.

Introduction to Oracle Discovery and Inventory

The term *Oracle discovery and inventory* refers to the process of examining a network to find and collect hardware and software inventory for every device with one or more Oracle applications installed on it. FlexNet Manager Suite performs software license compliance calculations on the collected inventory data to provide information about what software you own against what software you have installed. To collect detailed Oracle inventory, you must have FlexNet Manager for Oracle, a separately licensed option for FlexNet Manager Suite. Oracle discovery and inventory collection process involves the following sequence of steps:

1. Discovery of which devices on the network have Oracle software installed.

2. Collection of detailed discovery information, hardware and general software inventory, and Oracle inventory from the discovered devices.

3. Calculating to report the number of entitlements consumed against the number of licenses purchased.

You can use either of the following two components to collect Oracle inventory in FlexNet Manager Suite:

- The FlexNet inventory agent, and in particular its core inventory collection executable `ndtrack` (`ndtrack.exe` on Windows, and `ndtrack.sh` on UNIX). The FlexNet inventory agent is the optimal collection method as it not only collects Oracle inventory, but also collects hardware and other software inventory at the same time. The hardware data is important for correct calculation of Oracle processor license types. FlexNet inventory agent (`ndtrack`) always executes on the target Oracle server in any one of the following ways:

  - When installed locally (via FlexNet Manager Suite's adoption process, or other software deployment tools, or manually) on the Oracle server. The process of installing FlexNet inventory agent through FlexNet Manager Suite is called *adoption*, because the Oracle server has been 'adopted' into close inventory management with the locally-installed `ndtrack` executable.

  - Deployed outside FlexNet Manager Suite using third-party methods. You have to manage deployment and operation of FlexNet inventory agent using third-party methods.

  - Deployed manually on a shared network folder. You can manually deploy FlexNet inventory agent on a shared folder and setup a process to allow the execution of the agent on the desired Oracle server.

- The FlexNet Beacon, and in particular its core inventory collection component, FlexNet Beacon engine, can connect *directly* to an Oracle database (but not to other Oracle applications like E-Business Suite) and collect inventory information from it. This method, called 'direct' inventory gathering, is valuable, but will not gather sufficient information for calculating Oracle process license positions, and must be augmented by additional hardware inventory information (which may come from third-party tools).

FlexNet Manager Suite can also import inventory data using `.xlsx` or `.csv` file uploads.

This chapter explains all the Oracle discovery and inventory collection methods supported by FlexNet Manager Suite while describing when to use which method. You can use the information in this chapter to select an Oracle inventory collection method appropriate for your enterprise.

Oracle Inventory Collection Methods

FlexNet Manager Suite offers several methods for Oracle inventory collection. Each of these methods has a different way of working and involves different Flexera software components. The following topics provides a brief overview of each inventory collection method. The details of required privileges and components may help you to select the inventory collection method suitable for your enterprise.

Agent-Based Inventory Collection

You create a discovery and inventory rule to discover and adopt Oracle servers in your network. The process of installing FlexNet inventory agent is called adoption. Once the servers are adopted, the locally-installed FlexNet inventory agent (`ndtrack.exe` on Windows, and `ndtrack.sh` on UNIX) collects hardware, software, Oracle database, and Oracle options discovery and inventory. It creates a discovery file that includes discovered Oracle services. The agent inventory schedule (configured through the web interface) controls the discovery and inventory process. The collected information is sent to the appropriate inventory beacon. The inventory beacon sends this information to FlexNet Manager Suite.

Privileges required

The following privileges are required based on the operating system of the Oracle server:

- **For Windows**: For the adoption process, an account (either a Windows domain account, or a local account on the Windows server) is required with full access to the Windows Service Control Manager on the Oracle server (specifically, it must have the `SC_MANAGER_ALL_ACCESS` privilege). You must register this account in the secure Password Store on the appropriate inventory beacon. The local NT_Authority\SYSTEM account must be a member of the `ora_dba` database group in the Oracle security settings. In the `sqlnet.ora` file located in the `ORACLE_HOME\network\admin` directory, the `SQLNet.AUTHENTICATION_SERVICES` property must be set to `(NTS)`.

- **For UNIX**: For the adoption process, a local account is required that has `ssh` privileges. You must register this account in the secure Password Store on the appropriate inventory beacon. FlexNet inventory agent runs with sudo privileges and impersonates the first user in the `dba` Oracle database group.

Zero Touch Inventory Collection

You create a discovery and inventory rule to collect discovery and inventory information for the devices identified by the target. When the rule executes on inventory beacons, the inventory gathering component `ndtrack` (`ndtrack.exe` on Windows, and `ndtrack.sh` on UNIX) of FlexNet inventory agent is remotely executed on every Oracle server identified by the target definition. The `ndtrack` collects hardware, software, Oracle database, and Oracle options discovery and inventory. It creates a discovery file that includes discovered Oracle services. The collected information is sent to the appropriate inventory beacon which sends this information to FlexNet Manager Suite..

Privileges required

The following privileges are required based on the operating system of the Oracle server:

- **For Windows**: For executing FlexNet inventory agent remotely, an account (either a Windows domain account, or a local account on the Windows server) is required with full access to the Windows Service Control Manager on the Oracle server (specifically, it must have the SC_MANAGER_ALL_ACCESS privilege). You must register this account in the secure Password Store on the appropriate inventory beacon. The local NT_Authority\SYSTEM account must be a member of the `ora_dba` database group in the Oracle security settings. In the `sqlnet.ora`, the `SQLNet.AUTHENTICATION_SERVICES` property must be set to `(NTS)`.

- **For UNIX**: For FlexNet inventory agent remotely, a local account is required that has `ssh` privileges. You must register this account in the secure Password Store on the appropriate inventory beacon. FlexNet inventory agent (`ndtrack`) runs with sudo privileges and impersonates one of the accounts in the `dba` Oracle database group.

Direct Inventory Collection

You can use any of the following direct inventory collection methods. Each of these methods uses different ways to discovery Oracle database servers in your network.

- **Direct inventory with Oracle network discovery**: The inventory collection component of the inventory beacon (FlexNet Beacon engine) scans its assigned subnet to discover Oracle database servers and then connects *directly* to Oracle databases to collect inventory information. The collected information is sent to FlexNet Manager Suite

- **Direct Inventory using `tnsnames.ora`**: The inventory collection component of the inventory beacon (FlexNet Beacon engine) uses the `tnsnames.ora` file to connect *directly* to Oracle databases within its assigned subnet to collect discovery and inventory information. The collected information is sent to FlexNet Manager Suite

- **Direct inventory with manual creation of discovery devices**: The operator manually enters the listener and services information for each Oracle server through the web interface of FlexNet Manager Suite. Using the connection information, the inventory collection component of the inventory beacon (FlexNet Beacon engine) connects *directly* to Oracle databases within its subnet and collects inventory.

Privileges required

You need an account with read-only permissions on every Oracle database for all the tables and views needed for collecting Oracle inventory. You must record the credentials for this account in the secure Password Store on the appropriate inventory beacon.

Comparison of Inventory Collection Methods

The following table presents a comparison of different inventory collection methods available with FlexNet Manager Suite:

Feature	Agent-based	Zero touch	Direct- nw scan	Direct- tnsnames.ora	Direct- manual	Spreadsheet upload
Components Required						
FlexNet inventory agent	Y					
inventory beacon	Y	Y	Y	Y	Y	Y
Discovery and inventory rules	Y	Y	Y	Y	Y	
Compatible ODAC drivers			Y	Y	Y	
The `tnsname.ora` file				Y		
The OEM Adapter (optional)				Y		
Privileges Required						
For Windows: Account with full access to Windows Service Control Manager on Oracle server. The `NT_Authority\SYSTEM` account as a member of `ora_dba`	Y	Y				
For UNIX: Local account with `ssh` privilege	Y	Y				
An account with read-only permissions on every Oracle database for all the tables and views needed for collecting Oracle inventory			Y	Y	Y	
Collected Information						
General hardware and software inventory	Y	Y				Y
Oracle database inventory	Y	Y	Y	Y	Y	Y
Oracle application and engineering system inventory	Y	Y				Y
Oracle LMS data for audit	Y	Y	Y	Y	Y	
User Actions Required						
Create a discovery and inventory rule with the **Allow these targets to be adopted** option selected in the target definition.	Y					
Create a discovery and inventory rule, selecting the **Discover devices using network scan** and **Gather hardware and software inventory** options in the **General** section of the action definition. If you wish to target only Oracle database servers, modify the port list to contain ports used by the Oracle database servers.		Y				
Create a discovery and inventory rule with the following options selected in the action definition:			Y			

Feature	Agent-based	Zero touch	Direct- nw scan	Direct- tnsnames.ora	Direct- manual	Spreadsheet upload
• **Discover devices using network scan** option in the **General** section • **Discover Oracle database environments**, **Port scan**, (and/or) **SNMP scan**, and **Gather Oracle database environment inventory** options in the **Oracle discovery and inventory** section.						
Create a discovery and inventory rule with the following options selected in the action definition: • **Discover devices using network scan** option in the **General** section • **Discover Oracle database environments**, **TNS names file**, and **Gather Oracle database environment inventory** options in the **Oracle discovery and inventory** section.				Y		
• Manually create discovery device records with listener and services information for all Oracle servers. • Create a discovery and inventory rule with the **Gather Oracle database environment inventory** option selected in the **Oracle discovery and inventory** section of the action definition.					Y	
Schedule file uploads from the FlexNet Beacon.						Y

Components for Oracle Inventory Collection

Each of the different inventory collection methods described above may involve different software components within FlexNet Manager Suite. The number and types of software components involved in Oracle inventory collection depends on the selected inventory collection method. This section provides a brief overview of each of the software components involved in inventory collection.

FlexNet inventory agent

A software agent that may be installed on different kinds of computers to collect software and hardware inventory information for the device on which it is installed. It is also installed on each inventory beacon, where it can be run by remote execution from the target server, collecting zero touch inventory. It is a 32-bit executable that transforms the inventory information into an XML formatted (`.ndi`) file and uploads it to an inventory beacon. All locally-installed FlexNet inventory agents collect inventory according to the agent inventory collection schedule defined in the FlexNet Manager Suite user interface. When the inventory beacon collects zero touch inventory, it follows

the schedule set in the applicable rule. For more details about discovery and inventory rules, see **Discovery and Inventory Rules** in the online help.

FlexNet Beacon

You can install FlexNet Beacon on any recent Microsoft Windows server to make it operate as an inventory beacon, where it acts as a collection point for inventory and business information within the enterprise. One or more network subnets are assigned to an inventory beacon, and the inventory beacon collects data from the devices present within the assigned subnets:

• Where the FlexNet inventory agent has been locally installed on the target device, it collects discovery information, hardware and general software inventory along with Oracle inventory, and uploads collected data to the inventory beacon

• FlexNet inventory agent is also included within the installation of the inventory beacon, where it can collect data remotely through zero touch inventory.

The collected data is then uploaded to the central application server, where it is processed to update the database records representing the software and hardware assets owned by the enterprise. For more information about inventory beacons, see **What is an Inventory Beacon?** in the online help.

Oracle Enterprise Manager Adapter

The Oracle Enterprise Manager (OEM) adapter is a software component that provides an alternative or additional method of discovering Oracle databases in your computing estate. FlexNet Manager Suite requires discovery information before collecting inventory. Discovery (for Oracle databases) includes the collection of connection details to allow inventory gathering. The OEM adapter gathers the database connection details for all the databases managed by an instance of Oracle Enterprise Manager and formats them into a standard `tnsnames.ora` file. Each installation of OEM adapter can connect to only one instance of Oracle Enterprise Manager, saving the resulting `tnsnames.ora` file to a fixed location on the inventory beacon. Therefore, you need an inventory beacon and one installation of the OEM adapter for each instance of Oracle Enterprise Manager.

The `tnsnames.ora` file generated by the OEM adapter is used by the FlexNet Beacon engine as one possible discovery method for use with direct inventory gathering method to collect Oracle database inventory. If you decide to use direct inventory gathering but not to deploy the OEM adapter, you can also copy the standard `tnsnames.ora` manually from the Oracle server to the TNSNames repository on the appropriate inventory beacon.

tnsnames.ora

The `tnsnames.ora` file is an Oracle-standard configuration file that contains connection descriptors for the services running on an Oracle database. The connection descriptors contain the host name, protocol, service name, and port for each of the services running on an Oracle database. The `tnsnames.ora` file may be created in at least two ways:

• By default, each time the services configuration is updated, Oracle automatically saves updated connection information in a `tnsnames.ora` file stored on the Oracle server. The FlexNet inventory agent (`ndtrack`) uses this standard Oracle file as part of its discovery process when executing on the Oracle server (whether through adoption, or zero touch inventory gathering). This is the default behavior of the FlexNet inventory agent and requires no specific option on the web interface of FlexNet Manager Suite.

- A separate `tnsnames.ora` file may be generated by the OEM adapter, as described above, for use in direct inventory gathering (when you cannot use `ndtrack`).

The OEM adapter generates one `tnsnames.ora` file for all the databases managed by an instance of Oracle Enterprise Manager, and saves it in the appropriate folder on an inventory beacon. Alternatively, if you already have a `tnsnames.ora` file with details of all Oracle database instances in a subnet, you can copy this file to `%Program Data%\Flexera Software\Repository\TNSNames` folder on the inventory beacon to which the subnet is assigned.

Discovery and Inventory Rules

You can use a discovery and inventory rule to configure discovery and inventory processes that you choose to run from an inventory beacon. You also need a rule if you choose to install FlexNet inventory agent automatically on each of the discovered Oracle servers. A rule is a combination of one or more targets, an action, and a schedule. When a discovery and inventory rule executes, it identifies devices based on the target definition(s) and performs the action specified in the rule on those devices. The target adoption settings can be used to adopt the identified devices (install FlexNet inventory agent locally on the devices). The action properties determine the actions to perform and the targets' properties determine the devices on which to perform the action. For more information about discovery and inventory rules, see **Discovery and Inventory Rules** in the online help.

Prerequisites for Oracle Discovery and Inventory

To collect detailed Oracle inventory, you must have FlexNet Manager for Oracle, a separately licensed option for FlexNet Manager Suite. You need this license to perform:

- License management and compliance for Oracle Processor, Oracle Named User Plus, Oracle Application User, and other Oracle-specific licenses

- Detailed Oracle inventory of database options and Oracle E-Business Suite

- Inventory of engineered systems like Exadata, Exalogic, SuperCluster, and so on

- Inventory reconciliation for the Oracle license types stated above.

Without the FlexNet Manager for Oracle option, you can:

- Perform only discovery of Oracle software titles including Oracle databases

- Collect installer evidence data (basic software inventory)

- Collect Normalized server application inventory by product family and version (for example, Oracle DB 9i).

In addition to the FlexNet Manager for Oracle license, you need different privileges for different inventory collection methods and the administrative access to FlexNet Manager Suite.

- When using FlexNet inventory agent to collect Oracle inventory, you need the following privileges based on the operating system of the Oracle server:

- **On Windows**: To install FlexNet inventory agent, an account (either a Windows domain account, or a local account on the Windows server) is required with full access to the Windows Service Control Manager on the Oracle server (specifically, it must have the SC_MANAGER_ALL_ACCESS privilege). You must register this account in the secure Password Store on the appropriate inventory beacon.

- **On UNIX**: To install FlexNet inventory agent on UNIX, a local account with `ssh` privileges is required. You must register this account in the secure Password Store on the appropriate inventory beacon. Once installed, the FlexNet inventory agent runs according to the agent inventory schedule defined through the web interface.

- When using zero touch inventory collection on Windows, you need the following privileges based on the operating system of the Oracle server:

  - **On Windows**: An account (either a Windows domain account, or a local account on the Windows server) is required with full access to the Windows Service Control Manager on the Oracle server (specifically, it must have the SC_MANAGER_ALL_ACCESS privilege). You must register this account in the secure Password Store on the appropriate inventory beacon. The `ndtrack` relies on Windows authentication to connect to Oracle service. The `ndtrack` can collect inventory only if the `SQLNet.AUTHENTICATION_SERVICES` property is set to `(NTS)` in the `sqlnet.ora` file (located in the `ORACLE_HOME/network/admin` folder).

  - **On UNIX**: For the adoption process, a local account is required that has `ssh` privileges. You must register this account in the secure Password Store on the appropriate inventory beacon. FlexNet inventory agent runs with sudo privileges and impersonates (pretends to be) one of the accounts in the `ora_dba` Oracle database group. The `ndtrack` reads the ORACLE_HOME path from the `oratab` file and finds the `tnsnames.ora` file. The `tnsnames.ora` file is parsed to discover SID and ports for each Oracle service.

- When using FlexNet Beacon engine to collect inventory using any of the direct inventory collection, you need an account with read-only permissions on every Oracle database for all the tables and views needed for collecting Oracle inventory. You must record the credentials for this account in the secure Password Store on the appropriate inventory beacon. You also need the appropriate ODAC driver on each inventory beacon.

Verifying the ORADBA Group Membership

The ORADBA (typically named `ora_dba` on Windows, and `dba` on UNIX) is a database user group that may exist or can be created on an Oracle server. The `ndtrack` (in agent-based or zero touch inventory collection) runs using the NT AUTHORITY\SYSTEM identity in Windows and `root` on UNIX. For Windows, the NT AUTHORITY \SYSTEM must be a member of `ora_dba` group. For UNIX, `ndtrack` impersonates the first database user in the `dba` group. You can perform the following test to verify whether Oracle local OS authentication is adequately configured for the FlexNet inventory agent (in either agent-based or zero touch inventory collection) to gather Oracle inventory:
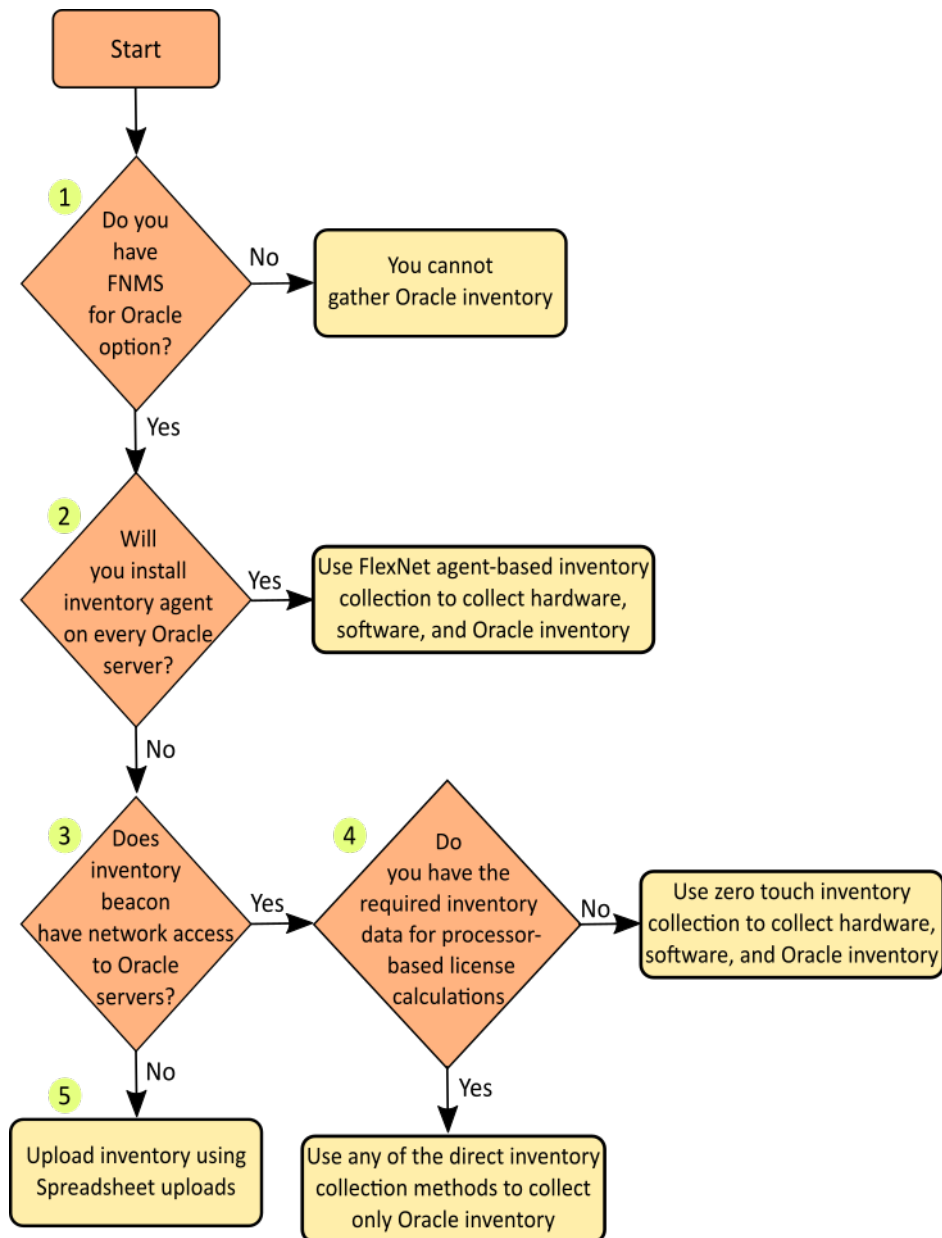
1. Log onto a computer running Oracle Database software as a user that is a member of the ORADB group.

2. Ensure that the ORACLE_HOME and ORACLE_SID environment variables are set.

3. Run the `sqlplus / as sysdba` command to test Oracle database connectivity.

4. Do either of the following:

   - **On Windows**: Verify that the local NT_Authority\SYSTEM account is a member of the ora_dba group (by default, this is the case)

- **On UNIX**: Verify that there is at least one user in the `dba` group. The FlexNet inventory agent runs as root, and then impersonates the first user in this group when connecting to Oracle database.

An Oracle installation in its default configuration should pass this test.

Selecting an Oracle Inventory Collection Method

The selection of a particular Oracle inventory gathering method depends on your software installation policies and the decisions made by your network and Oracle database administrators. The choice and usage of Flexera inventory gathering components depends on your network and database access policies. For example, you may prefer either to deploy FlexNet inventory agent on each of your Oracle servers, or to collect inventory from one or more inventory beacons (zero touch inventory). The following diagram provides a broad overview for selection of an Oracle inventory collection method. The details of each inventory collection method are discussed on the following pages. Numbers in the diagram correspond to the description below:

```
                          ┌──────────┐
                          │  Start   │
                          └──────────┘
                               │
                               ▼
                         ╱────────────╲
                   ①    ╱   Do you     ╲
                       ╱     have        ╲      No      ┌──────────────────────┐
                      ╱      FNMS         ╲────────────▶│   You cannot          │
                       ╲    for Oracle   ╱              │ gather Oracle inventory│
                        ╲   option?     ╱               └──────────────────────┘
                         ╲────────────╱
                               │ Yes
                               ▼
                         ╱────────────╲
                   ②    ╱    Will      ╲
                       ╱  you install   ╲     Yes      ┌──────────────────────────┐
                      ╱ inventory agent  ╲────────────▶│ Use FlexNet agent-based inventory │
                      ╲ on every Oracle  ╱             │ collection to collect hardware,   │
                       ╲   server?      ╱              │ software, and Oracle inventory    │
                         ╲────────────╱                └──────────────────────────┘
                               │ No
                               ▼
                         ╱────────────╲                ╱────────────╲
                   ③    ╱   Does       ╲          ④  ╱    Do        ╲
                       ╱  inventory     ╲            ╱  you have the  ╲    No    ┌──────────────────────┐
                      ╱   beacon        ╲  Yes      ╱ required inventory╲──────▶│ Use zero touch inventory│
                      ╲ have network access╱───────▶╲ data for processor-╱      │ collection to collect hardware,│
                       ╲  to Oracle      ╱           ╲ based license    ╱       │ software, and Oracle inventory │
                        ╲  servers?     ╱             ╲ calculations   ╱        └──────────────────────┘
                         ╲────────────╱                ╲────────────╱
                               │ No                          │ Yes
                   ⑤          ▼                              ▼
              ┌──────────────────────┐         ┌──────────────────────────┐
              │ Upload inventory using│         │ Use any of the direct inventory │
              │ Spreadsheet uploads   │         │ collection methods to collect   │
              └──────────────────────┘         │ only Oracle inventory           │
                                               └──────────────────────────┘
```

1. To perform license management and compliance for Oracle-specific licenses, you need a license for the FlexNet Manager for Oracle option. Without this option, you can only discover Oracle infrastructure and collect basic Oracle software inventory.

2. If you can install a FlexNet inventory agent on each of the Oracle servers, use a discovery and inventory rule to adopt Oracle servers and use the **Discovery & Inventory** > **Settings** option on the web interface to adjust the global agent inventory collection schedule. Whenever this job is triggered, each of the locally-installed FlexNet inventory agents collects the inventory for its device, and uploads the results to the appropriate inventory beacon. Flexera Software recommends this method of Oracle discovery and inventory. For more information, see *How Does Agent-Based Inventory Collection Work* on page 76.

3. If you choose not to install a FlexNet inventory agent on each of the Oracle servers, you can collect inventory through either zero touch or direct inventory collection method. In zero touch inventory collection, FlexNet Manager Suite remotely executes `ndtrack` (installed on the inventory beacon or a shared network drive) on every Oracle server and collects discovery information, hardware and general software inventory, and Oracle inventory for all discovered devices. The collected information is sent to the appropriate inventory beacon. You can configure a discovery and inventory rule to define which devices to target for inventory collection and what action to perform on those targets. The inventory collection is carried out according to the schedule of the underlying rule.

4. If you need only Oracle database inventory and you have detailed hardware inventory (required for calculating Oracle processor-based licenses), you can use any of the direct inventory collection methods. In any of the *direct* inventory collection methods, the inventory gathering component of the inventory beacon uses existing discovery information (ports, the `tnsnames.ora` file, or discovery information from manually created devices) to establish a direct connection to each Oracle database and gathers only Oracle inventory. Remember that the direct inventory methods can collect inventory only for Oracle databases.

5. If you have a detailed inventory generated through third-party systems, or you cannot establish a network connection between an inventory beacon and the Oracle servers, you can use spreadsheet inventory uploads to import Oracle inventory into FlexNet Manager Suite. FlexNet Manager Suite supports the import of inventory information formatted in either comma-separated value (`.csv`) files or Microsoft Excel spreadsheets (`.xlsx`) files. You can download the fixed templates from FlexNet Manager Suite user interface, populate your Oracle inventory data, and schedule uploads of the inventory spreadsheets through FlexNet Beacon.

How to Gather Oracle Inventory

This section marks the end of the planning phase for Oracle inventory collection and assumes that you have selected the most suitable method to implement. It provides detailed information about how each of the Oracle inventory collection methods works, with a detailed procedure for collecting Oracle inventory through each of the available methods.

How Does Agent-Based Inventory Collection Work

In agent-based inventory collection, FlexNet Manager Suite collects Oracle inventory through a FlexNet inventory agent installed on each Oracle server within your network. When the agent inventory collection job (scheduled from the web interface) executes, each FlexNet inventory agent gathers discovery information, hardware and general software inventory, and Oracle inventory for the adopted device. It transforms the gathered information into the following files and uploads them to the appropriate inventory beacon:

• A `.disco` file for discovered devices

• An `.ndi` file for hardware and software inventory

• An `.ndi` file for Oracle inventory.

The inventory beacon uploads all collected discovery and inventory information to the central application server. This method is simplest in operation, as each FlexNet inventory agent performs discovery and inventory for its respective Oracle server.

You create a discovery and inventory rule to discover and adopt Oracle servers in your network. The process of installing FlexNet inventory agent is called adoption. Once the servers are adopted, the FlexNet inventory agents collect inventory information based on the agent inventory schedule. The following diagram shows an example scenario with only one inventory beacon. Most FlexNet Manager Suite installations involve the installation of multiple inventory beacons within your network.
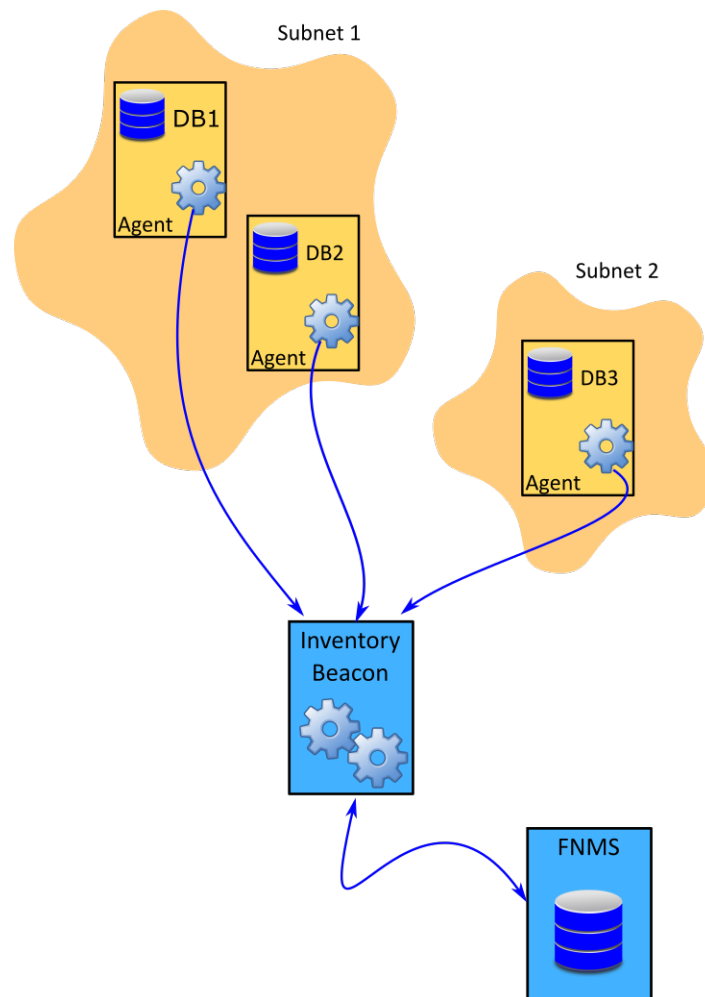
The diagram shows three database servers, two in Subnet1 and one in Subnet2. An instance of FlexNet inventory agent has been installed on each of these servers. The inventory beacon has been assigned to cover both of these subnets and can connect to every Oracle server. The following process describes how settings in rules flow down to the inventory beacons, trigger installation of the FlexNet inventory agent where required, and initiate inventory collection and upload.

Agent-Based Inventory Collection Process

The agent-based inventory collection process is the easiest method of inventory collection. In this fully-automated method, FlexNet Manager Suite adopts each of the Oracle servers by installing and configuring a copy of FlexNet inventory agent on it. The `ndtrack` finds and parses the tnsnames.ora file to get the SID (a database identifier) for each Oracle service running on the adopted Oracle server. Once installed locally on the Oracle server, FlexNet inventory agent collects discovery and inventory information using one of the following methods:

- On Windows, it runs using the NT_Authority\SYSTEM and relies on Windows authentication to connect to Oracle service. The `ndtrack` can collect inventory only if the property `SQLNet.AUTHENTICATION_SERVICES` is set to `(NTS)` in the `sqlnet.ora` file (located in the `ORACLE_HOME \network\admin` folder).

- On UNIX, it uses a local account that has `ssh` privileges. This account must be registered this account in the secure Password Store on the appropriate inventory beacon. After adoption, the FlexNet inventory agent runs with sudo privileges and impersonates (pretends to be) one of the accounts in the `dba` Oracle database group to collect discovery and inventory.

The collected discovery and inventory information is uploaded to the appropriate inventory beacon. The following are the steps to gather Oracle inventory using this method:

1. Make sure you have the required prerequisites for this type of inventory collection. For more information, see *Prerequisites for Oracle Discovery and Inventory* on page 72 and *Oracle Inventory Collection Methods* on page 67.

2. If not already done, deploy and configure one or more inventory beacon(s) in your network by navigating to **Discovery & Inventory** > **Beacons**. For detailed information about inventory beacons, their deployment, and configuration, see the topics under **What Is an Inventory Beacon?** and **Inventory Beacon** in the online help.

3. Verify the operational status of each inventory beacon by checking the following properties on the **Discovery & Inventory** > **Beacons** page:>

Property	Expected Value
Beacon status	`Operating normally`
Policy status	`Up to date`
Connectivity status	`Connected`

4. Ensure that your organizational site and subnet hierarchy is recorded through the **Discovery & Inventory** > **Subnets** page on the user interface. For more information on creating and managing subnets, see **Subnets** in the online help.

5. Ensure that you have assigned the defined subnets to the deployed and configured inventory beacons through the **Discovery & Inventory** > **Beacons** page on the user interface. For more information, see **Assigning a Subnet to a Beacon** in the online help.

6. If not already done, create and configure an account to access the database.

    - **For Windows**: Create an account (either a Windows domain account, or a local account on the Windows server) with full access to the Windows Service Control Manager on the Oracle server (specifically, it must have the SC_MANAGER_ALL_ACCESS privilege). You must register this account in the secure Password

Store on the appropriate inventory beacon. The local SYSTEM account must be a member of the `ora_dba` database group in the Oracle security settings. Ensure that adequate credentials are available for the automated installation process to run by recording them in the secure Password Store available on each inventory beacon. For more information, see the online help for the inventory beacon.

- **For UNIX**: For the purpose of FlexNet inventory agent installation, create a local user account and assign `ssh` privileges to it for the purpose of FlexNet inventory agent installation. Register this account in the secure Password Store on the appropriate inventory beacon. After installation, FlexNet inventory agent runs with sudo privileges and impersonates one of the accounts in the `dba` Oracle database group.

Important • The locally-installed FlexNet inventory agent cannot collect Oracle inventory for the Oracle server without an `ora_dba` user group. If your database administrator will not allow an `ora_dba` group, you can use direct or spreadsheet uploads inventory collection methods. You can also execute `ndtrack` using a specific DBA user. For details, see Appendix D - Inventory Collection With Specific DBA on page 111.

7. To install FlexNet inventory agent on each of the discovered servers, create a discovery and inventory rule with the following details. For more information about discovery and inventory rules, see **Discovery and Inventory Rules** in the online help.

- **Target**: Navigate to **Discovery & Inventory** > **Discovery and Inventory rules** and click the **Target** tab. Create a target to identify all Oracle servers in your network. You can use subnet name, IP address, or device name pattern to identify devices in the target definition. Select the **Allow these targets to be adopted** option. For more information on targets, see **Targets** in the online help.

- **Action**: Create an action with one or more discovery options selected in the **Action settings** section of the **Create an Action** page. An action can **Discover devices using a network scan** or **Microsoft Windows Computer Browser service**. Select the **Discover devices** and **Gather hardware and software inventory** options the action definition. For more information about creating and managing actions, see **Actions** in the online help.

- **Schedule**: Specify when this rule should run. In general, you can set the frequency to `Once`.

Note • The purpose for running this discovery and inventory rule is to discover Oracle servers and deploy FlexNet inventory agent on each of them. Once all your Oracle servers are adopted, the inventory collection is managed through the agent inventory schedule.

8. The rule flows down to the inventory beacons with the next policy update. An inventory beacon updates its policy after every 15 minutes by default. For more information, see the **Inventory Settings** page in the online help.

9. The inventory beacon receives a policy update and passes the updated `InventorySettings.xml` file to each FlexNet inventory agent. The FlexNet inventory agents use this file to update the **Agent Inventory Schedule**. This schedule determines the frequency of inventory collection from the adopted Oracle servers.

10. Based on the operating system of the Oracle server, FlexNet inventory agent uses one of the following methods to collect discovery and inventory information:

- **On Windows**: The account that you created in step 6 is used to create a service that runs `ndtrack` using the `SYSTEM` account. The `SYSTEM` account is a member of the `ora_dba` database group. The `ndtrack` reads the ORACLE_HOME path from the `HKLM\SOFTWARE\Oracle\` and `HKLM\SOFTWARE\Wow6432Node\Oracle\` location in the registry and finds the `tnsnames.ora` file. The `tnsnames.ora` file is parsed to discover SID for each Oracle service.

  Note • The `ndtrack` relies on Windows authentication to connect to Oracle service. The `ndtrack` can collect inventory only if the property `SQLNet.AUTHENTICATION_SERVICES` is set to `(NTS)` in the `sqlnet.ora` file (located in the `ORACLE_HOME\network\admin` folder)

- **On UNIX**: The account that you created in step 6 is used to run `ndtrack` with sudo privileges and it impersonates the first user in the `dba` database group to access the database. The `ndtrack` reads the ORACLE_HOME path from the `oratab` file and finds the `tnsnames.ora` file. The `tnsnames.ora` file is parsed to discover SID for each Oracle service.

11. The FlexNet inventory agent sends the discovery (`.disco` files) and inventory (`.ndi` files) information to the appropriate inventory beacon.

12. The inventory beacon uploads this information to FlexNet Manager Suite.

13. Navigate to **Discovery & Inventory** > **Discovery and Inventory Rules** and click the rule name to view its status. Wait until the **Status** field shows `Completed`. This process may take some time to complete and you may have to revisit or refresh the page from time to time.

14. Check the value of the **Devices adopted** and **Devices already adopted** columns in the **Adoption results** and ensure that all your Oracle databases have been discovered.

   Tip • The rule does not discover devices that are powered off at the time of rule execution. To adopt such undiscovered Oracle servers, run the rule once again.

15. If the **Status** column displays `Completed with errors`, check the adoption status for errors. For more information, see *Troubleshooting Oracle Discovery* on page 97.

16. Wait for the license reconciliation process to start. Alternatively, you can start this process manually by navigating to **License Compliance** > **Reconcile**. To update inventory before the compliance calculation, ensure that the **Update inventory for reconciliation** check box is selected. The uploaded FlexNet inventory is saved to the FlexNet inventory database. An automated process imports data from the FlexNet inventory database to the central compliance database. Once started, this task appears on the **System Tasks** page. This process may take some time to finish and you may have to revisit or refresh the page from time to time.

17. Navigate to the **Discovery & Inventory** > **Oracle Instances** page. You should see your Oracle servers listed on this page.

Note • You can also collect inventory if there is no network connection between the FlexNet inventory agent and appropriate inventory beacon. See Appendix C - Inventory Collection when Inventory Beacon is Disconnected from FlexNet Agent on page 110.

How Does Zero Touch Inventory Collection Work

In zero touch inventory collection, you create a discovery and inventory rule with one or more targets and an action to collect discovery and inventory information. When the rule executes, the inventory gathering component `ndtrack` (`ndtrack.exe` on Windows, and `ndtrack.sh` on UNIX) of FlexNet inventory agent (installed on either the inventory beacon or on a shared network location) is remotely executed on every Oracle server identified by target definition. The `ndtrack` collects discovery information, hardware and general software inventory, and Oracle inventory for each Oracle server. The collected information is transformed into the following files and uploaded to the appropriate inventory beacon:

- A `.disco` file for discovered devices

- An `.ndi` file for hardware and software inventory

- An `.ndi` file for Oracle inventory.

The inventory beacon uploads all collected discovery and inventory information to the central application server. The following diagram shows an example scenario:
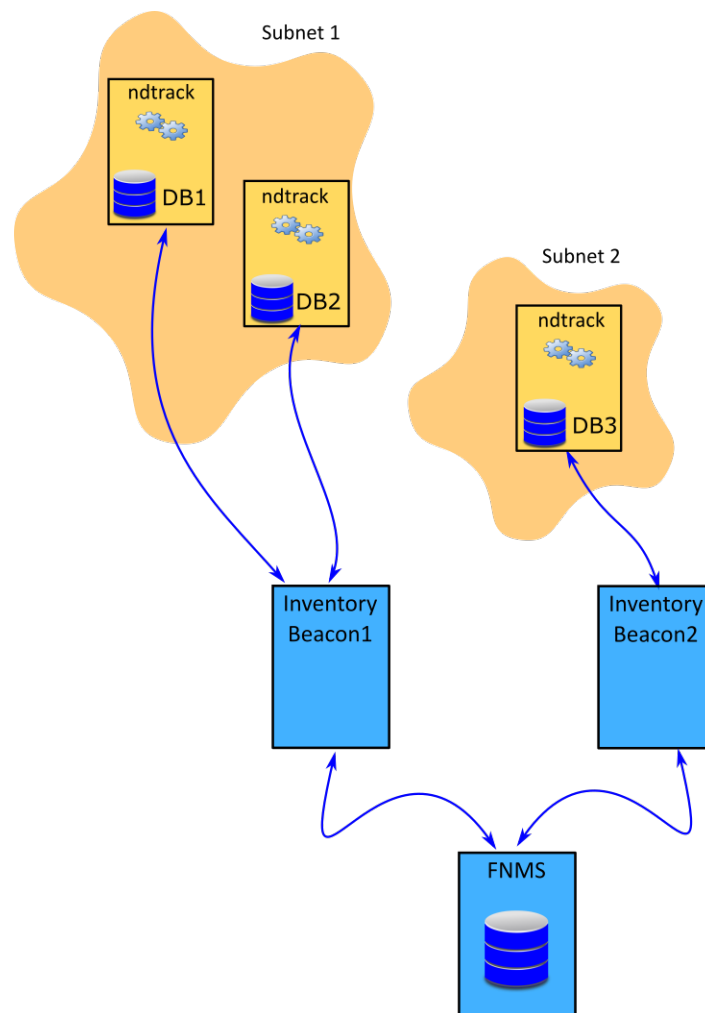
The above diagram shows three database servers, two on Subnet1 and one on Subnet2. The Subnet1 is assigned to Inventory Beacon1 and Subnet2 is assigned to Inventory Beacon2. When a discovery and inventory rule (that has these database servers identified as targets) runs, `ndtrack` (installed on inventory beacon1) is remotely executed on DB1 and DB2 to collect discovery and inventory of DB1 and DB2 servers. Similarly, `ndtrack` (installed on inventory beacon2) is remotely executed on DB3 to collect discovery and inventory of DB3. The inventory beacons upload the collected data to the central application server of FlexNet Manager Suite. The following process describes how settings in rules flow down to the inventory beacons and initiate discovery and inventory collection, and upload.

Zero-Touch Inventory Collection Process

You can use the zero touch inventory collection process to collect Oracle inventory when you cannot install FlexNet inventory agent on every Oracle server. In zero touch inventory collection, you create a discovery and inventory rule to collect discovery and inventory information. When the rule executes, the inventory gathering component `ndtrack` (`ndtrack.exe` on Windows, and `ndtrack.sh` on UNIX) of FlexNet inventory agent (installed on either inventory beacon or a shared network location) is remotely executed on every Oracle server

identified by target definition. The remote execution of `ndtrack` varies with the operating system of the Oracle server:

- On Windows, it runs as an account with full access to Windows Service Control Manager and uses Windows authentication to connect to Oracle services. The `NT AUTHORITY\SYSTEM` must be a member of `ora_dba` group.

- On UNIX, it uses a local account that has `ssh` privileges. This account must be registered this account in the secure Password Store on the appropriate inventory beacon. The FlexNet inventory agent runs with sudo privileges and impersonates the first user in the `dba` Oracle database group.

The collected discovery and inventory information is uploaded to the appropriate inventory beacon. Following are the steps to gather Oracle inventory using this method:

1. Make sure you have the required prerequisites for this type of inventory collection. For more information, see *Prerequisites for Oracle Discovery and Inventory* on page 72 and *Oracle Inventory Collection Methods* on page 67.

2. If not already done, deploy and configure one or more beacons in your network by navigating to **Discovery & Inventory** > **Beacons**. For detailed information about inventory beacon, its deployment, and configuration, see the topics under **What Is an Inventory Beacon** and **inventory beacon** in the online help.

3. Verify the operational status of each inventory beacon by checking the following inventory beacon properties on the **Discovery & Inventory** > **Beacons** page:

Property	Expected Value
Beacon status	`Operating normally`
Policy status	`Up to date`
Connectivity status	`Connected`

4. Ensure that your organizational site and subnet hierarchy is recorded through the **Discovery & Inventory** > **Subnets** page on the user interface. For more information on creating and managing subnets, see **Subnets** in the online help.

5. Ensure that you have assigned the defined subnets to the deployed and configured inventory beacons through the **Discovery & Inventory** > **Beacons** page on the user interface. For more information, see the **Assigning Subnets to a beacon** topic in the online help.

6. If not already done, create and configure an account to access the database.

- **For Windows**: Create an account (either a Windows domain account, or a local account on the Windows server) with full access to the Windows Service Control Manager on the Oracle server (specifically, it must have the SC_MANAGER_ALL_ACCESS privilege). You must register this account in the secure Password Store on the appropriate inventory beacon. The local `NT_Authority\SYSTEM` account must be a member of the `ora_dba` database group in the Oracle security settings. Ensure that adequate credentials are available for the automated remote execution to run by recording them in the secure Password Store available on each inventory beacon. For more information, see the online help for the inventory beacon.

- **For UNIX**: Create a local user account and assign `ssh` privileges to it. Register this account in the secure Password Store on the appropriate inventory beacon. The FlexNet inventory agentruns with sudo privileges and impersonates one of the accounts in the `dba` Oracle database group.

Important • *The* `ndtrack` *cannot collect Oracle inventory for the Oracle server without an* `ORADB` *user group. If your database administrator will not allow an* `ora_dba` *group, you can use direct or spreadsheet uploads inventory collection methods. You can also execute* `ndtrack` *using a specific DBA user. For details, see* *Appendix D - Inventory Collection With Specific DBA on page 111.*

7. To discover Oracle servers and collect inventory, navigate to **Discovery & Inventory** > **Discovery and Inventory rules** and create a discovery and inventory rule with the following details. For more information about discovery and inventory rules, see **Discovery and Inventory Rules** in the online help.

   • **Target**: Create a target to identify all Oracle servers in your network. You can use subnet name, IP address, or device name matching pattern to identify devices in the target definition. For more information on targets, see the **Targets** page in the online help.

   • **Action**: Create an action and select **Discover devices using network scan** and **Gather hardware and software inventory** options in the **General** section of action definition. Ensure that the correct port numbers have been entered.

   • **Schedule**: Specify the running schedule for this rule.

8. The rule flows down to the inventory beacons with the next beacon policy update. An inventory beacon updates its policy after every 15 minutes by default. For more information, see the **Inventory Settings** page in the online help.

9. Based on the operating system of the Oracle server, `ndtrack` uses one of the following methods to collect discovery and inventory information:

   • **On Windows**: The account that you created in step 6 is used to create a service that runs `ndtrack` using the `SYSTEM` account. The `NT_Authority\SYSTEM` account is a member of the `ora_dba` database group. The `ndtrack` reads the ORACLE_HOME path from the `HKLM\SOFTWARE\Oracle\` and `HKLM\SOFTWARE\Wow6432Node\Oracle\` location in the registry and finds the `tnsnames.ora` file. The `tnsnames.ora` file is parsed to discover SID and ports for each Oracle service.

   Note • *The* `ndtrack` *relies on Windows authentication to connect to Oracle service. The* `ndtrack` *can collect inventory only if the property* `SQLNet.AUTHENTICATION_SERVICES` *is set to* `(NTS)` *in the* `sqlnet.ora` *file (located in the* `ORACLE_HOME\network\admin` *folder).*

   • **On UNIX**: The account that you created in step 6 is used to run `ndtrack` runs with sudo privileges and impersonates the first user in the `dba` database group to access the database. The `ndtrack` reads the ORACLE_HOME path from the `oratab` file and finds the `tnsnames.ora` file. The `tnsnames.ora` file is parsed to discover SID and ports for each Oracle service.

   Note • *It is also possible to run the* `ndtrack` *executable manually on the Oracle server, using any account with administrator privileges and membership in the* `ora_dba` *group.*

10. Navigate to **Discovery & Inventory** > **Discovery and Inventory Rules** and click the rule name to view its status. Wait until the **Status** field shows `Completed`. This process may take some time to complete and you may have to revisit or refresh the page from time to time.

11. Check the value of the **Service discovered**, **Inventory completed**, **Inventory skipped**, and **Inventory failed** columns in the **Current run** and ensure that discovery and inventory has been collected for all your Oracle servers.

Tip • The rule does not discover devices that are powered off at the time of rule execution. To discover and collect inventory from such Oracle servers, rerun the discovery and inventory rule.

12. If the **Status** column displays `Completed with errors`, use the troubleshooting information to resolve the errors. For more information, see the topics under *Troubleshooting Oracle Inventory Collection* on page 93.

13. If the **Status** column displays `Completed`, wait for the license reconciliation process to start. Alternatively, you can start this process manually by navigating to **License Compliance** > **Reconcile**. To update inventory before the compliance calculation, ensure that the **Update inventory for reconciliation** check box is selected. The uploaded FlexNet inventory is saved to the FlexNet inventory database. An automated process imports data from the FlexNet inventory database to the central compliance database. Once started, this task appears on the **System Tasks** page. This process may take some time to finish and you may have to revisit or refresh the page from time to time.

14. Navigate to the **Discovery & Inventory** > **Oracle Instances** page. You should see your Oracle servers listed on this page.

Note • You can deploy FlexNet inventory agent on a shared location or local file system in your network and run it from each Oracle server to collect discovery and inventory information. For more information, see Appendix B - Deploying FlexNet Inventory Agent on a Shared Location on page 108. You can also collect inventory if there is no network connection between the FlexNet inventory agent and appropriate inventory beacon. See Appendix C - Inventory Collection when Inventory Beacon is Disconnected from FlexNet Agent on page 110.

How Does Direct Inventory Collection Work

In direct inventory collection, the inventory collection component of the inventory beacon (FlexNet Beacon engine) connects directly to each of the targeted Oracle databases within its assigned subnet using discovery information from any of the following sources, and collects *only* Oracle database inventory:

• **Port scan**: Enables FlexNet Beacon engine to connect to Oracle database using the specified ports. FlexNet Manager Suite adds 1521 and 2483 as default ports. You can change the default ports or add more as your environment requires.

- **SNMP scan**: Enables FlexNet Manager Suite to connect to Oracle database using Simple Network Management Protocol (SNMP).

- **TNS Names file**: Enables FlexNet Beacon engine to connect to Oracle databases using the `tnsnames.ora` configuration file present in the TNSNames repository folder on the inventory beacon. The default path for this repository is `%ProgramData%\Flexera Software\Repository\TNSNames`. This file may have been copied from Oracle, or generated by the OEM adapter.

- **Gather Oracle database environment inventory**. This option uses the listener and services information from the manually created discovered devices and directly collects database inventory. No other discovery option is required with this option.

*Note • You can select the discovery source from the **Oracle discovery and inventory** section in the action settings.*

The collected Oracle inventory is uploaded to FlexNet Manager Suite.

To collect discovery and inventory information, you define a discovery and inventory rule with one or more targets and an action. The target specifies where to look for Oracle servers (for example, subnet address or host name) and the action specifies how to discover Oracle servers (for example, using the `tnsnames.ora` file) and what action to perform on them. When the rule executes, it instructs each FlexNet Beacon engine to connect to Oracle databases and collect Oracle inventory if the targeted devices are within its assigned subnet. Each FlexNet Beacon engine establishes a direct connection to the Oracle database using ODAC (Oracle Data Access Components) drivers and gathers inventory information. You can use your Oracle support account to access the ODAC driver compatibility matrix from `http://metalink.oracle.com/metalink/plsql/ml2_documents.showDocument?p_database_id=NOT&p_id=207303.1`. Another version of the compatibility matrix (maintained by a third-party consulting organization) is available at `http://www.dba-oracle.com/t_oracle_client_versions_higher_lower_database_release.htm`. The following diagram shows an example scenario:
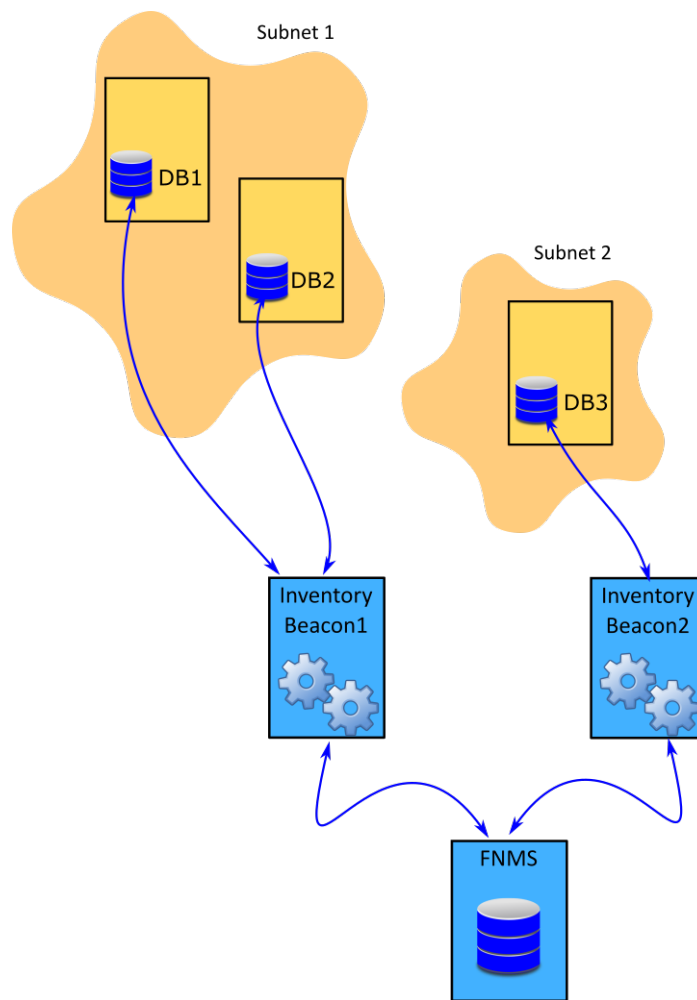
The above diagram shows three database servers, two on Subnet1 and one on Subnet2. The Subnet1 is assigned to Inventory Beacon1 and Subnet2 is assigned to Inventory Beacon2. When a discovery and inventory rule (that has these database servers identified as targets) runs, FlexNet Beacon engine (from inventory beacon1) directly connects to DB1 and DB2 and collects Oracle inventory. Similarly, FlexNet Beacon engine (from inventory beacon2) directly connects to DB3 and collects Oracle inventory. The inventory beacons upload the collected data to the central application server FlexNet Manager Suite.

Oracle Discovery by Using OEM Adapter

You can also use the OEM adapter as an alternative or additional method of discovering Oracle databases in your computing estate. The OEM adapter collects connection data from Oracle Enterprise Manager server, formats it into a `tnsnames.ora` file, and saves this file on a special location on the inventory beacon. An instance of OEM adapter can connect to only one instance of Oracle Enterprise Manager. The following diagram shows an example scenario:
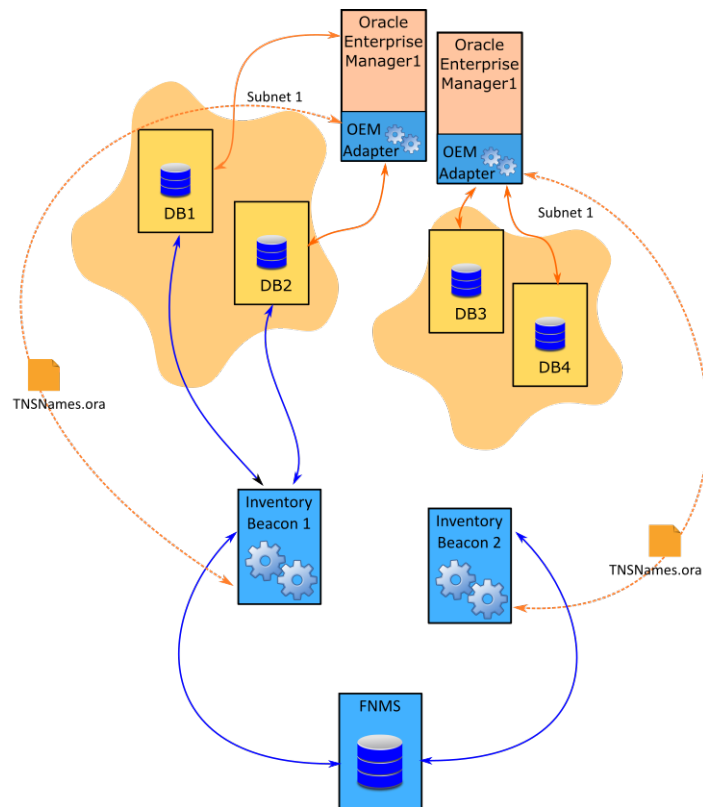
The above diagram shows that Subnet1 has been assigned to Inventory Beacon 1 and it contains two databases, DB1 and DB2 managed by OEM1. Similarly, Subnet2 has been assigned to inventory beacon2 and it contains two databases DB3 and DB4, managed by OEM2. In this example, each OEM adapter:

1. Connects to the corresponding OEM server and collects the connection information for all Oracle servers managed by the OEM server.

2. Formats this information into a `tnsnames.ora` file.

3. Places this file in the TNSNames repository folder on the corresponding inventory beacon.

Note • When some (not all) of your Oracle servers are managed by Oracle Enterprise Manager, the installed OEM adapter generates the `tnsnames.ora` file with information of servers that are managed by OEM. For the Oracle servers that are not managed by OEM, you can manually create a `tnsnames.ora` file, rename it to (for example, `Manualtnsnames.ora`), and copy it to the `%Program Data%Flexera Software\Repository \TNSNames` folder on the inventory beacon. In this case, you must change the name of the manually created file because the next run of the OEM adapter overwrites the `tnsnames.ora` file in the TNSNames repository. For more information on installing and using the OEM adapter, please see FlexNet Manager Suite Adapters Reference.

The FlexNet Beacon engine component of each inventory beacon uses the `tnsnames.ora` file saved locally on the inventory beacon by the OEM adapter to directly connect to the identified Oracle databases that are within its assigned subnet.

Direct Inventory Collection Process

In direct inventory collection method, the FlexNet Beacon engine component of the inventory beacon connects directly to every Oracle server within its assigned subnets and collects only Oracle inventory. The following are the steps to gather Oracle inventory using this method:

1. Make sure you have the required prerequisites for this type of inventory collection. For more information, see *Prerequisites for Oracle Discovery and Inventory* on page 72 and *Oracle Inventory Collection Methods* on page 67.

2. Ensure that you have deployed and configured one or more beacons in your network by navigating to **Discovery & Inventory** > **Beacons**. For detailed information about the inventory beacon, its deployment, and configuration, see the topics under **What Is an Inventory Beacon?** and **Inventory Beacons** in the online help.

3. Ensure that your organizational site and subnet hierarchy is recorded through the **Discovery & Inventory** > **Subnets** page on the user interface. For more information on creating and managing subnets, see **Subnets** in the online help.

4. Ensure that you have assigned the defined subnets to the deployed and configured inventory beacons through the **Discovery & Inventory** > **Beacons** page on the user interface. For more information, see **Assigning a Subnet to a Beacon** in the online help.

5. Verify the operational status of each inventory beacon by checking the following inventory beacon properties on the **Discovery & Inventory** > **Beacons** page:

Property	Expected Value
Beacon status	`Operating normally`
Policy status	`Up to date`
Connectivity status	`Connected`

6. Install the appropriate 32-bit Oracle Provider for OLEDB on each inventory beacon that accesses Oracle servers. Please read the supported platform details, and download the appropriate driver. FlexNet Manager Suite uses the ODAC driver from this package.

Note • *the FlexNet Beacon engine can collect inventory from an Oracle server only if it has a locally-installed ODAC driver compatible with the database version on the Oracle server. You can install only one ODAC driver on an inventory beacon. If a subnet requires multiple ODAC drivers, you need to install and configure multiple inventory beacons. To look at this another way, if a subnet has multiple Oracle servers that require different ODAC drivers, you need to split the subnet.*

7. Set up a special account with read-only permissions on your Oracle database for all the tables and views needed for collecting Oracle inventory. see *Appendix A- Oracle Tables and Views for Oracle Inventory Collection* on page 106. One helpful practice is to use the same set of credentials on all servers. This makes it easier to register a single set of credentials in the Password Stores on all applicable inventory beacons, and to script creation of the account consistently across your Oracle servers. Flexera Software provides a script to create and configure this database user. To get this script, log into the Flexera Software Knowledge Base (*https://flexeracommunity.force.com/customer/CCKnowledgeBase*, or access through the Support pages of the company website), and search for article Q200934. For details about Oracle tables and views required for inventory collection, see *Appendix A- Oracle Tables and Views for Oracle Inventory Collection* on page 106.

*Note • The sole purpose of creating this audit user is to collect Oracle inventory. However, FlexNet Manager Suite counts it as a named user while calculating license compliance for Oracle licenses. You can adjust the license consumption for this user to avoid any license compliance failure. Navigate to the **Oracle Instance Properties** > **Oracle users** page and set the consumption for this user to zero. For more information, see Oracle Users Tab.*

8. Record the credentials for the special account in the secure Password Store available on each inventory beacon. The FlexNet Beacon uses this information to connect to Oracle listeners using the configured ports in the discovery and inventory rule. For more information, see the online help for the inventory beacon.

9. To discover Oracle servers and collect inventory, navigate to **Discovery & Inventory** > **Discovery and Inventory rules** and create a discovery and inventory rule with the following details. For more information about discovery and inventory rule, see **Discovery and Inventory Rules** in the online help.

   - **Target**: Create a target to identify all Oracle servers in your network. You can use subnet name, IP address, or device name matching pattern to identify devices in the target definition. For more information on targets, see the **Targets** page in the online help.

   - **Action**: Create an action and specify the source of discovery that each inventory beacon should use to connect to Oracle databases for inventory collection. The following table describes the required action settings for available discovery options:

Discovery Method	Action Settings
Discovery using Network Scan	• **Discover Oracle databases using network scan** option in the **Oracle discovery and inventory** section • Specify the database port numbers.
Discovery using `tnsnames.ora`	**Discover Oracle databases** and **TNS names file** options in the **Oracle discovery and inventory** section. *Note • You can manually create the `tnsnames.ora` file or copy one generated by Oracle, or generate it through the OEM adapter. With direct inventory collection, FlexNet Beacon collects inventory only for those Oracle*

Discovery Method	Action Settings
	servers that are both within the subnet assigned to this inventory beacon, and identified in the `tnsnames.ora` file.
Direct inventory Gathering with manual creation of discovery devices	**Gather Oracle databases environment inventory** option selected in the **Oracle discovery and inventory** section. This option needs no other discovery option. When you select this option, each FlexNet Beacon uses the database listener and service information of manually created discovered devices to establish a direct connection. *Note • This method requires you to manually create discovery device records with listener and services information for all Oracle servers. For more information, see* **Create a Discovery Device** *in the online help*

Note • Some Oracle license calculations are dependent on the hardware information of the host. To get accurate license compliance calculations, Flexera Software recommends you to select the following options with any of the above discovery methods. This would trigger general software and hardware inventory collection for Oracle servers through `ndtrack`*.*

- *Discover devices using network scan or Discover devices using Microsoft Windows Computer Browser service*

- *Gather hardware and software inventory.*

- *Schedule: Specify the running schedule for this rule.*

10. The rule flows down to the inventory beacons with the next beacon policy update. An inventory beacon updates its policy after every 15 minutes by default. For more information, see the **Inventory Settings** page in the online help.

11. Navigate to **Discovery & Inventory** > **Discovery and Inventory Rules** and click the rule name to view its status. Wait until the **Status** field shows `Completed`. This process may take some time to complete and you may have to revisit or refresh the page from time to time.

12. Check the value of the **Service discovered**, **Inventory completed**, Inventory skipped, and Inventory failed columns in the **Current run** and ensure that discovery and inventory has been collected for all your Oracle servers.

Tip • The rule does not discover devices that are powered off at the time of rule execution. To discover and collect inventory from such Oracle servers, rerun the discovery and inventory rule.

13. If the **Status** column displays `Completed with errors`, use the troubleshooting information to resolve the errors. For more information, see the topics under *Troubleshooting Oracle Inventory Collection* on page 93.

14. If the **Status** column displays `Completed`, wait for the license reconciliation process to start. Alternatively, you can start this process manually by navigating to **License Compliance** > **Reconcile**. To update inventory before the compliance calculation, ensure that the **Update inventory for reconciliation** check box is selected. The uploaded FlexNet inventory is saved to the FlexNet inventory database. An automated process imports data from the FlexNet inventory database to the central compliance database. Once started, this task appears on the **System Tasks** page. This process may take some time to finish and you may have to revisit or refresh the page from time to time.

15. Navigate to the **Discovery & Inventory** > **Oracle Instances** page. You should see your Oracle servers listed on this page.

How Does Spreadsheet Upload Work

To collect inventory using agent-based, zero touch, or direct inventory collection methods, each inventory beacon must be able to access Oracle servers present in its assigned subnets. You may not be able to use any of these methods if your security practices do not allow a network connection between Oracle servers and the corresponding inventory beacons.

You can use the scheduled upload of spreadsheets of Oracle inventory data from the FlexNet Beacon user interface to upload data. This option periodically takes spreadsheets from a defined location and uploads them to FlexNet Manager Suite.

When you use the same named connection to import a spreadsheet file into FlexNet Manager Suite from second time onwards, FlexNet Manager Suite:

• Updates the changes to existing records

• Inserts new records

• Deletes the previously saved records that are no longer present in the latest upload

• De-duplicates incoming data rows.

For each named connection, imported spreadsheet data is preserved on the central application server, and re-imported for each consumption calculation. This has the following implications:

• Having multiple connection names for the same (or overlapping) data set(s) is poor practice. Since the spreadsheet data for each connection is preserved centrally, and imported in an undefined order, data may toggle between the values in different spreadsheets.

• While data that disappears from a sole import source is also removed from the operations databases, data in overlapping data sources is retained until it disappears from all sources.

Important • FlexNet Manager Suite treats each spreadsheet connection as a different data source, and each of these spreadsheet connections is imported into FlexNet Manager Suite without a set order. If you create two spreadsheet connections with same records but different data values, data may toggle from the value in one spreadsheet to the value in another. It is strongly recommended to keep only one named spreadsheet connection

for each data set. Also, if you are uploading a spreadsheet using a new connection, the previously saved records that are no longer present in the latest upload, are retained. The existing records with changed data in the new upload, are updated. This way, a superset of data records is created when you upload data using different connections. For example, if the first upload from Connection1 uploads 100 devices and another upload from Connection2 uploads 100 devices (50 of them were also a part of the upload done through Connection1), the resultant data set in FlexNet Manager Suite will have 150 records.

The following is the brief overview of inventory collection using this method. For more details, see **Managing Inventory Spreadsheet Connections** in the online help. Also, the *Importing Inventory Spreadsheets and CSV Files* chapter of the *FlexNet Manager Suite System Reference* guide has a detailed coverage of spreadsheet uploads.

1. From the FlexNet Beacon interface on any inventory beacon, select the spreadsheet type and template.

2. Populate data into the template manually or through scripts.

3. Setup the connection for uploading templates. This includes setting the Connection Name, Connection Folder, and Overlapping Inventory Filter settings.

4. Schedule the connection execution to upload the spreadsheet data to FlexNet Manager Suite.

5. The inventory upload triggers the license reconciliation process. Once started, this task appears on the **System Tasks** page.

6. Wait for the license reconciliation process to complete.

The uploaded data appears on the **Oracle Instances** page.

Troubleshooting Oracle Inventory Collection

In a system with complex network architecture and Oracle RAC (Real Application Cluster), you may sometimes need to troubleshoot your Oracle inventory collection process. For zero touch and direct inventory collection methods, you can use the **Rules** page as a starting point for troubleshooting. This page dynamically displays the status of each task and step resulting from a discovery and inventory rule execution. The information on this page enables you to identify the problems and the troubleshooting method required to resolve those problems. In contrast, to troubleshoot agent-based discovery and inventory, you have to scan the system logs to identify what went wrong as the **Rules** page only displays the adoption status. This section contains the detailed troubleshooting procedures for Oracle discovery and inventory collection.

Troubleshooting Discovery and Inventory Rules

The discovery and inventory rule is the first line of troubleshooting Oracle discovery and inventory collection using zero touch and direct inventory collection methods. You should begin troubleshooting discovery and inventory rule if the **Rules** or **Rule Execution Details** pages indicates any problems related to rules. For example, the `Discovery shows that this device is an Oracle database server but the rule does not`

`allow for Oracle database inventory gathering` message on the **Rule Execution Details** page indicates a problem with the rule. This page mainly displays the following information:

- **Discovery information**: Devices targeted, discovered, skipped, and failed

- **Adoption information**: Devices targeted, adopted, skipped, and failed

- **Inventory information**: Services discovered, inventory completed, skipped, and failed.

The most common indicators for discovery and inventory rules troubleshooting are when:

- No inventory is returned at all

- Inventory is missing for a particular type of software, such as for Oracle database

- Devices in a particular subnet do not return inventory

- You find no inventory (or incomplete inventory) for a particular device

- One or more Oracle servers are not adopted.

The following are the steps to troubleshoot discovery and inventory issues caused by improper rule settings:

1. Navigate to **Discovery & Inventory** > **Discovery and Inventory Rules** and expand the rule record.

2. Ensure that the **Rule status** field displays `Enabled`. A disabled rule never executes.

3. Ensure that the rule has been correctly scheduled.

4. Edit the rule and open the associated target(s). Ensure that the target covers the subnet or device for which the inventory information was expected. If you are using this rule for agent-based inventory collection, ensure that **Allow these targets to be adopted** is selected in the target definition.

5. Ensure that the expected device or subnet is not excluded from the target(s). When a rule has multiple targets, the exclusions always override an inclusion. For example, if a device is excluded in a target definition, and the same device is included in some other target definition, the rule would exclude that device.

6. Edit the rule components to ensure that the appropriate options are selected for the selected inventory collection method:

Inventory collection method	Required actions

Agent-based	- Ensure the following in the target definition:
	
	  - The desired site or subnet is included
	
	  - The **Allow these targets to be adopted** option is selected.
	
	- At least one discovery option is selected in the **General devices discovery and inventory** section of the action definition.

Zero touch	Select the following options in the **General devices discovery and inventory** section of the action definition: - **Discover devices using network scan** and/or **Discover devices using Microsoft Windows Computer Browser service.**

Inventory collection method	Required actions
	• **Gather hardware and software inventory**.
Direct inventory using network scan	Select the following options in the action definition: • **Discover devices using network scan** and **Gather hardware and software inventory** options in the **General** section • **Discover Oracle database environments**, **Port scan**, and **Gather Oracle database environment inventory** options in the **Oracle discovery and inventory** section.
Direct inventory using `tnsnames.ora`	Select the following options in the action definition: • **Discover devices using network scan** and **Gather hardware and software inventory** options in the **General** section • **Discover Oracle database environments**, **TNS names file**, and **Gather Oracle database environment inventory** options in the **Oracle discovery and inventory** section.
Direct inventory manual creation of discovery devices	Select the **Gather Oracle database environment inventory** option in the **Oracle discovery and inventory** section of the action definition.

7. Save the changes and rerun the rule.

8. Navigate to **Discovery & Inventory** > **Discovery and Inventory Rules** and expand the rule record. It displays the details of the current and the last run of the rule.

9. Check the rule **Status** to see the execution status of this rule. You may have any of the following values:

   • `In Progress`: The rule execution is in progress. You should wait until the next status change. You may need to refresh this page from time to time.

   • `Completed`: The rule execution completed successfully. This indicates that no further troubleshooting is required.

   • `Completed with errors`: The rule execution encountered some errors. You should proceed with troubleshooting procedures.

10. Click the **Show**/**hide task status and history** link to view the status of the tasks generated by this rule.

11. Click the + icon next to the rule name to view the inventory beacons involved in the execution of this rule. You can also expand the inventory beacon record to view the steps performed by the FlexNet Beacon. The **Summary** column displays a summary of the steps executed as a part of the rule execution.

12. Notice the status of each step performed by the FlexNet Beacon. For example, if the **Performing discovery** step shows **Completed**, it indicates that the discovery process has been completed successfully by FlexNet Beacon on this inventory beacon.

13. Check the summary information for each of the steps to identify a possible problem. For example, if the Oracle discovery completed successfully, you will see the total number of Oracle database servers discovered. If the number of discovered devices doesn't match the expectation, there may be a problem with the discovery process.

Note • A hyphen (-) after a discovery entry indicates that the particular discovery is not selected as a part of the action. For example, the following output indicates that the underlying action should be modified to do Oracle database discovery. See Troubleshooting Oracle Discovery on page 97 for details.

```
Oracle database servers discovered: -
```

14. Check the summary information for the `Gathering Oracle database inventory` step to identify the problems with inventory. For example, a non-zero number in `Devices failed to be inventoried` indicates a problem with the inventory process.

15. FlexNet Manager Suite records system level activities for discovery and inventory tasks in the following log files on every inventory beacon. You can click the **Download log** link to download the log file specific to the step that generated one or more errors on this inventory beacon. You may use these file for the advanced troubleshooting procedures explained in the following pages.

- `adoption.log` for adoption-specific errors

- `Discovery.log` for general discovery information

- `DeviceInventory.log` for hardware and software inventory

- `OracleDBInventory.log` for Oracle inventory.

16. You can click the **See details** link to view the **Rule Execution Details** page. The **Summary** field indicates the problems encountered during the execution of this rule. For more details, see **Rule Execution Details** in the online help.

Troubleshooting Adoption

If you are using agent-based Oracle discovery and inventory collection, a successful adoption of an Oracle server is a prerequisite to collect Oracle inventory for the server. You will not get Oracle discovery and inventory information for the Oracle server where FlexNet inventory agent fails to install. The following are the steps to troubleshoot the adoption of Oracle servers:

1. Check the **Rules** page to see the number of devices targeted, already adopted, adopted, skipped, and failed. You may narrow down your troubleshooting to failed or skipped devices.

2. If the adoption failed on the Oracle server, check the following log files on the Oracle server for any adoption-specific errors.

- `adoption.log` in the `%temp%` folder on Windows

- `ndinstlr.log` and `ndinstlrsh.log` in the `/var/tmp/flexera/log/` directory on UNIX.

3. If you are unable to resolve the adoption-specific errors, contact Flexera Software Support with the log files.

Troubleshooting Oracle Discovery

Discovery, in one way or the other, is a prerequisite for gathering inventory. A problem with an Oracle device discovery may lead to a missing inventory record on the **Oracle Instances** page. If a particular Oracle device record is missing from this page (after the inventory collection and license reconciliation is over), you should investigate for possible problems with the discovery and inventory process. You should start with basic troubleshooting procedure followed by advanced troubleshooting (specific to the discovery method used). This section lists the basic troubleshooting steps for Oracle discovery. See the following topics for discovery method-specific troubleshooting.

1. Navigate to **Discovery & Inventory** > **All Discovered Devices**.

2. Click the filter icon and set a filter `Oracle= Yes`.

3. Click the flag icon to view the device records with errors.

4. Click the **Name** link to open the device record, click the **Status** tab, and expand the **Oracle database inventory** section. This displays the error message as returned by the listener.

5. Try to resolve the error with help from Oracle DBA. The following is a list of common Oracle listener errors with suggested resolutions:

Error Message	Possible Resolutions
ORA-12170: TNS: Connect timeout occurred	• Ensure that no firewall is blocking connection to the database • Ensure that the database is online and running on correct IP.
ORA-12541: TNS: no listener	• The database is shutdown. Ask your database administrator to start the database • Ensure that no firewall is blocking connection to the database • Ensure that the listener is not password protected. If it is, add the listener password to the password store on the appropriate inventory beacon.
ORA-00942: table or view does not exist	Ensure that the table listed in the error message exists. Ask your database administrator to create the table if it does not exist.
ORA-12514: TNS: listener does not currently know of service requested in connect descriptor	TNS names file is not correct. This is common when a database is set to listen for only "SID" or only "Service_Name". For more information, see **<Troubleshooting direct inv>**
ORA-01017: invalid username/password; logon denied	Indicates incorrect permissions. Ask your Oracle database administrator to rerun the Flexera Permission script. For more information, see *Direct Inventory Collection Process* on page 89 and the troubleshooting topic specific to the discovery method used.

Error Message	Possible Resolutions
ORA-01034: ORACLE not available ORA-27101: shared memory realm does not exist Linux-x86_64 Error: 2: No such file or directory	• Ensure that the `oratab` file exists at its default location. • Ensure that the `tnsname.ora` file exists in the TNS Names repository folder.
ORA-01033: ORACLE initialization or shutdown in progress	Oracle is stuck in a reboot process. Ask your database administrator to shutdown and restart the database.
ORA-12518: TNS: listener could not hand off client connection	Indicates a network problem. Ensure that the appropriate inventory beacon can access the Oracle server.
ORA-00604: error occurred at recursive SQL level 1	Indicates incorrect permissions. Ask your Oracle database administrator to rerun the Flexera Permission script. For more information, see *Direct Inventory Collection Process* on page 89 and the troubleshooting topic specific to the discovery method used.
ORA-00257: archiver error. Connect internal only, until freed	Indicates an archive error. Ask your Oracle database administrator to check the `archiver.log` file for the error.

Before you proceed to advanced troubleshooting methods, you should check if FlexNet inventory agent has been installed on the Oracle server. If the agent is not installed, you need to investigate the discovery method used for the discovery and inventory collection. Follow these steps to identify the discovery method:

1. Navigate to **Discovery & Inventory** > **All Discovered Devices** and verify the value of the **Agent installed** column for the Oracle host record. The value `Yes` indicates that FlexNet inventory agent has been installed on the device.

2. Navigate to **Discovery & Inventory** > **Discovery and Inventory Rules** and expand the rule record.

3. Identify the action name from the rule details.

4. Open the action and check the **Action settings** section to identify the discovery method selected.

5. Before using the appropriate troubleshooting procedure from the following topics, use the basic troubleshooting methods explained earlier in *Troubleshooting Discovery and Inventory Rules* on page 93.

Troubleshooting Agent-Based Discovery

The locally-installed FlexNet inventory agent prepares a `.disco` file needed for FlexNet Manager Suite to complete the appropriate discovered device records. It does this by reading the `listener.ora` files and interrogating Oracle listeners. The resultant `.disco` file along with its inventory (`.ndi`) files is uploaded to the appropriate inventory beacon. FlexNet Manager Suite does not display any status information for agent-based discovery. Check the log files directly to troubleshoot discovery issues. Follow these steps to troubleshoot discovery using FlexNet inventory agent:

1. Ensure that the device is adopted. See *Troubleshooting Adoption* on page 96.

2. Verify the contents of the Discovery folder. The `ndtrack` places the generated discovery file in this folder, before it is uploaded to the appropriate inventory beacon, and deleted from this folder. If this folder is empty, verify the Oracle server discovery record on the **All Discovered Devices** page.

   • **On Windows**: `%ProgramData%\ManageSoft Corp\ManageSoft\Common\Uploads\Discovery`

   • **On UNIX**: `/var/opt/managesoft/uploads/discovery/` directory.

3. The absence of a discovery device record for the Oracle server on the **All Discovered Devices** page or presence of a `.disco` file in this folder indicates some issue with discovery file upload. Check the `upload.log` file in the `%temp%\ManageSoft\` folder on Windows and in `/var/opt/managesoft/log` directory on UNIX for issues related to the upload.

4. Verify the following events from `tracker.log`. The `ndtrack` generates this log in the `%temp%\ManageSoft\` folder on Windows and in `/var/opt/managesoft/log` directory on UNIX.

   • The `listener.ora` file is discovered and processed. If `listener.ora` is not discovered, ensure that the file is present at its default location, which is `%ORACLE_HOME\network\admin` on Windows and `$ORACLE_HOME/network/admin` on UNIX. If the `listener.ora` file is not present at its default location, enable file scan on the directory where this file is present.

   • The event `Uploading file <filename.disco>` indicates the generation of the discovery file.

   • The event `file<filename.disco> removed from upload directory` indicates the successful upload of the discovery file.

5. If `tracker.log` provides no information about discovery completion, carry out the following procedure to verify the discovery file generation:

   a. Enable tracing by removing # from the following lines of the `etcp.trace` file present in the FlexNet inventory agent install folder on the Oracle server:

   ```
   +Inventory/Oracle
   +Inventory/Oracle/SDK
   +Inventory/Oracle/Query
   +Inventory/Oracle/Query/Substitution
   +Inventory/Oracle/Query/Execution
   +Inventory/Oracle/Listener
   +Inventory/Oracle/Listener/Detail
   ```

   b. Use the following command on the Oracle server to perform discovery without uploading discovery file to the inventory beacon. This enables you to verify the creation of the discovery file on the Oracle server.

      • **For Windows**: `Start ndtrack -t machine -o Upload=False`

      • **For UNIX**: `bin/sh ./ndtrack.sh -t machine -o Upload=False`

   c. Open the `%ProgramData%\ManageSoft Corp\ManageSoft\Common\Uploads\Discovery` folder on the Oracle server and open the discovery file to verify data about Oracle services. If there is no data related to Oracle services, open the `managesoft.log` file under the following locations and verify the status of Oracle discovery:

      • **On Windows**: `C:\windows\Temp\`

      • **On UNIX**: `/var/opt/flexera/`

    d. Try to resolve the indicated errors.

    e. Open the `%ProgramData%\Flexera Software\Incoming\Discovery` folder on the appropriate inventory beacon and look for the discovery file containing the device name of the Oracle server.

6. If the problem persists, contact Flexera Software Support with the log files.

7. Check the server side `uploader.log` in the `C:\Windows\Temp\ManageSoft` folder on the inventory beacon for errors in uploading data from inventory beacon to FlexNet Manager Suite.

8. Wait for the next discovery and inventory collection job to complete and verify the discovered device records on the **All Discovered Devices** page. This process may take some time to finish and you may have to revisit or refresh the page from time to time. If the problem persists, contact Flexera Software support with log files.

Troubleshooting Discovery Via Zero Touch Inventory Collection

The FlexNet inventory agent (through remote execution) prepares a `.disco` file needed for FlexNet Manager Suite to complete the appropriate discovered device records. It does this by reading the `listener.ora` files and interrogating Oracle listeners. The resultant `.disco` file along with its inventory (`.ndi`) files is uploaded to the appropriate inventory beacon. Most of the troubleshooting information for this method is displayed on the **Rules** page. Follow these steps to troubleshoot network discovery issues:

1. Check the **Rules** page to see the number of devices targeted, discovered, skipped, and failed. You can narrow down your troubleshooting to undiscovered devices.

2. If you are using the **Discover devices using network scan option**, verify the ports settings specified in the **General devices discovery and inventory** section of the action definition. If you are not sure about the correct port settings, ask your system administrator.

3. Ensure that the **Gather hardware and software inventory** option is selected.

4. Expand the rule record on the **Rules** page and click **Show/hide task status and history**.

5. Look for the **Gathering hardware and software inventory** step with a status `Completed with errors`. Download the corresponding log file to know more about the problem. For example, if the log indicates a problem with authentication, configure an account for database access. See *Zero-Touch Inventory Collection Process* on page 82.

6. Check the server side `uploader.log` in the `C:\Windows\Temp\ManageSoft` or `%temp%` folder on the inventory beacon for errors in uploading data from inventory beacon to FlexNet Manager Suite.

7. Enable tracing by removing # from the following lines of the `etcp.trace` file present in the `%Program Files (x86)%\FlexNet Manager Platform\Inventory Beacon\` folder:

```
+Inventory/Oracle
+Inventory/Oracle/SDK
+Inventory/Oracle/Query
+Inventory/Oracle/Query/Substitution
+Inventory/Oracle/Query/Execution
+Inventory/Oracle/Listener
+Inventory/Oracle/Listener/Detail
```

8. Wait for the next discovery and inventory collection job to complete, and verify the discovered device records on the **All Discovered Devices** page. If the problem persists, contact Flexera Software support with log files.

Troubleshooting Discovery for Direct Inventory Using Network Scan

The FlexNet Beacon engine connects directly to Oracle databases using database port information. Follow these steps to troubleshoot discovery:

1. Check the **Rules** page to see the number of devices targeted, discovered, skipped, and failed. You may narrow down your troubleshooting to undiscovered devices.

2. If you are using the **Discover devices using network scan option**, verify the ports settings specified in the **General devices discovery and inventory** section of the action definition. If you are not sure about the correct port settings, ask your system administrator.

3. Ensure that a special account to access the database has been configured and its credentials are recorded in the secure Password Store available on each inventory beacon. For details, see *Direct Inventory Collection Process* on page 89 and *Appendix A- Oracle Tables and Views for Oracle Inventory Collection* on page 106.

4. Verify the ports settings specified in the **Oracle discovery and inventory** section of the action definition. If you are not sure about the correct port settings, ask your database administrator. For more information, see *Prerequisites for Oracle Discovery and Inventory* on page 72.

5. If the Oracle listener is password protected, make sure to add the listener account in the Password Store on every inventory beacon.

6. Expand the rule record on the **Rules** page and click **Show/hide task status and history**. Look for the `Performing discovery` step with a status `Completed with errors`. Download the corresponding log file to know more about the problem.

7. Check the server side `uploader.log` in the `C:\Windows\Temp\ManageSoft` folder on the inventory beacon for errors in uploading data from inventory beacon to FlexNet Manager Suite.

8. Enable tracing by removing # from the following lines of the `etcp.trace` file present in the `%Program Files (x86)%\FlexNet Manager Platform\Inventory Beacon\` folder:

```
+Inventory/Oracle
+Inventory/Oracle/SDK
+Inventory/Oracle/Query
+Inventory/Oracle/Query/Substitution
+Inventory/Oracle/Query/Execution
+Inventory/Oracle/Listener
+Inventory/Oracle/Listener/Detail
```

9. Try to resolve the discovered errors and rerun the rule to collect Oracle discovery and inventory information.

10. Wait for the next discovery and inventory collection job to complete and verify the discovered device records on the **All Discovered Devices** page. If the problem persists, contact Flexera Software support with log files.

Discovery Using tnasname.ora

The FlexNet Beacon engine connects directly to Oracle databases using discovery information from the `tnsnames.ora` file. Follow these steps to troubleshoot discovery:

1. Check the **Rules** page to see the number of devices targeted, discovered, skipped, and failed. You may narrow down your troubleshooting to undiscovered devices.

2. Ensure that you have selected the **TNS names file** option in the action definition. For more information, see **Creating an Action** in the online help.

3. Verify the existence of `.ora` file(s) in the TNS names repository folder on every inventory beacon involved. The location of the TNSNames repository is `%ProgramData%\Flexera Software\Repository\TNSNames` folder.

4. If the Oracle listener is password protected, make sure to add the listener account in the Password Store on every inventory beacon.

5. Expand the rule record on the **Rules** page and click **Show/hide task status and history**. Look for the `Performing discovery` step with a status `Completed with errors`. Download the corresponding log file to know more about the problem.

6. Check the server side `uploader.log` in the `C:\Windows\Temp\ManageSoft` folder on the inventory beacon for errors in uploading data from inventory beacon to FlexNet Manager Suite.

7. Enable tracing by removing # from the following lines of the `etcp.trace` file present in the `%Program Files (x86)%\FlexNet Manager Platform\Inventory Beacon\` folder:

```
+Inventory/Oracle
+Inventory/Oracle/SDK
+Inventory/Oracle/Query
+Inventory/Oracle/Query/Substitution
+Inventory/Oracle/Query/Execution
+Inventory/Oracle/Listener
+Inventory/Oracle/Listener/Detail
```

8. Try to resolve the discovered errors and rerun the rule to collect Oracle discovery and inventory information.

9. Wait for the next discovery and inventory collection job to execute and verify the discovered device records on the **All Discovered Devices** page. If the problem persists, contact Flexera Software support with log files.

Discovery for Direct Inventory With Manual Creation of Discovery Devices

The FlexNet Beacon engine uses the discovery information from existing devices. No troubleshooting is required for discovery using this method.

Troubleshooting Oracle Inventory

After the discovery information is collected, FlexNet Manager Suite attempts to collect inventory information. Before you start troubleshooting Oracle inventory collection, you should check if FlexNet inventory agent has been installed on the Oracle server. If the agent is not installed, you need to investigate the discovery method used for the discovery and inventory collection, and select the appropriate troubleshooting procedure. Follow these steps to identify the discovery method:

1. Navigate to **Discovery & Inventory** > **All Discovered Devices** and verify the value of the **Agent installed** column for the Oracle host record. The value `Yes` indicates that FlexNet inventory agent has been installed on the device.

2. Navigate to **Discovery & Inventory** > **Discovery and Inventory Rules** and expand the rule record.

3. Identify the action name from the rule details.

4. Open the action and check the **Action settings** section to identify the discovery method selected.

5. Before using the appropriate troubleshooting procedure from the following topics, use the basic troubleshooting methods explained earlier in *Troubleshooting Discovery and Inventory Rules* on page 93.

Troubleshooting Agent-Based Inventory

For agent-based inventory collection to work, you need the `ora_dba` user group on every adopted Oracle server. If this group is not available, ask your database administrator to create one for you. For more details, see *Agent-Based Inventory Collection Process* on page 78. FlexNet Manager Suite only displays no information about inventory collection from the adopted devices on the **Rules** page. Check the log files directly to troubleshoot discovery issues. Follow these steps to troubleshoot agent-based inventory collection:

1. Check the **Rules** page to see information about the targeted devices that were adopted, skipped, and failed.

2. Ensure that the `InventorySettings.xml` in the `%ProgramData%\Flexera Software\Beacon \InventorySettings` folder on the adopted device has Oracle information. This file contains LMS scripts that are used to query Oracle applications and databases. You can verify the LMS scripts by finding the following code line in the `InventorySettings.xml`:

   - **On Windows**: `<RecognitionRule Id="OracleRule" Clasification="Client" Enabled="1" Name="Oracle" Platform="Windows" Type="Inherit" BaseRecognitionRuleId="OracleQuery" LicenseTerms="PurlBladeOracle">`

   - **On UNIX**: `<RecognitionRule Id="OracleRule" Clasification="Client" Enabled="1" Name="Oracle" Platform="Unix" Type="Inherit" BaseRecognitionRuleId="OracleQuery" LicenseTerms="PurlBladeOracle">`

3. If the LMS information is not present:

   a. Download the oracle information from `http://<FNMS server> /inventory-beacons/api/ download/inventory-settings` for single tenant.

> *Note •* *For managed service providers using multi-tenant mode, use* `http://<FNMS server> /` `inventory-beacons/api/download/inventory-settings?tenantUid=<tenantID>.`

    b. Place the file in `%ProgramData%\Flexera Software\Beacon\InventorySettings` folder.

    c. Rename it to add a `.zip` extension (`inventory-settings.zip`).

    d. Extract the file and rerun the discovery and inventory rule.

4. Verify the following in the `tracker.log` produced by `ndtrack` on the Oracle server. This log is present in the `%temp%\ManageSoft\` folder on Windows and in `/var/opt/managesoft/log` directory on UNIX.

- The events `Starting oracle inventory` and `Finished generating inventory` indicate the start and end of the Oracle inventory collection process

- The event `Uploading file <`*`filename`*`(Oracle).ndi.gz>` indicates the upload of the Oracle inventory

- The event `file <`*`filename`*`.(Oracle).ndi.gz> removed from upload directory` indicates the successful upload of the inventory file.

5. Navigate to **License Compliance** > **Reconcile** and ensure that the **Update inventory for reconciliation** option is selected. Wait for the next reconciliation job to complete. You can check the job status on the **System Tasks** page. This process may take some time to complete and you may have to revisit or refresh the page from time to time. Verify the Oracle server records on the **Oracle Instances** page.

6. If the problem persists, contact Flexera Software Support with the appropriate log files.

Troubleshooting Zero Touch Inventory

Most of the troubleshooting information for zero touch inventory collection is available on the **Rules** page in the web interface for FlexNet Manager Suite. Before using the troubleshooting methods stated in this section, implement the methods stated in *Troubleshooting Discovery and Inventory Rules* on page 93.

Following are the steps to troubleshoot zero touch Oracle inventory gathering:

1. Check the action definition to verify that you have selected the required inventory collection options.

2. Check the **Rules** page to see information about the targeted devices that were adopted, skipped, and failed.

3. Expand the rule record on the **Rules** page and click **Show/hide task status and history**. Look for the `Gathering Oracle inventory` step with a status `Completed with errors`. Download the corresponding log file to know more about the problem. For example, if the log indicates a problem with authentication, configure the account for database access.

4. The **Devices failed to be inventoried** field indicates the number of devices for which the inventory collection has failed. You can click the link to view the details in the **Rule Execution Details** page. For more information, see the **Rule Execution Details** page in the online help.

5. Check the `DeviceInventory.log` and `OracleDBInventory.log` file in the `%ProgramData%\Flexera Software\Compliance\Logging\InventoryRule` folder on the inventory beacon to see inventory-specific errors.

6. Navigate to **License Compliance** > **Reconcile** and ensure that the **Update inventory for reconciliation** option is selected. Wait for the next reconciliation job to complete. You can check the job status on the **System Tasks** page. This process may take some time to complete and you may have to revisit or refresh the page from time to time. Verify the Oracle server records on the **Oracle Instances** page.

7. If the problem persists, download the related log files from the web interface and contact Flexera Software Support.

Troubleshooting Direct Inventory

Most of the troubleshooting information for direct inventory collection is available on the **Rules** page in the web interface for FlexNet Manager Suite. Before using the troubleshooting methods stated in this section, implement the methods stated in *Troubleshooting Discovery and Inventory Rules* on page 93.

Following are the steps to troubleshoot Oracle inventory gathering using this method:

1. Ensure that you install the appropriate Oracle Data Access Components (ODAC) driver on each inventory beacon that accesses Oracle servers. For details, see *Zero-Touch Inventory Collection Process* on page 82

2. Make sure to set up a special account with read-only permissions on every Oracle database for all the tables and views needed for collecting Oracle inventory. Record the credentials for the special account in the secure Password Store available on each inventory beacon. For details about the required tables and views, see *Appendix A- Oracle Tables and Views for Oracle Inventory Collection* on page 106.

3. Expand the rule record on the **Rules** page and click **Show/hide task status and history**. Look for the `Gathering Oracle inventory` step with a status `Completed with errors`. Download the corresponding log file to know more about the problem.

4. The **Devices failed to be inventoried** field indicates the number of devices for which the inventory collection has failed. You can click the link to view the details in the **Rule Execution Details** page. For more information, see the **Rule Execution Details** page in the online help.

5. Check the `DeviceInventory.log` and `OracleDBInventory.log` file in the `%ProgramData%\Flexera Software\Compliance\Logging\InventoryRule` folder on the inventory beacon to see inventory-specific errors.

6. Navigate to **License Compliance** > **Reconcile** and ensure that the **Update inventory for reconciliation** option is selected. Wait for the next reconciliation job to complete. You can check the job status on the **System Tasks** page. Verify the Oracle server records on the **Oracle Instances** page.

7. If the problem persists, download the related log files from the web interface and contact Flexera Software Support.

Troubleshooting Spreadsheet Uploads

A spreadsheet upload by a FlexNet Beacon automatically triggers a compliance import job that is visible on the **System Tasks** page on the web interface of FlexNet Manager Suite. The **Log** column contains a link to the **Inventory Upload Validation Errors** page. You can click this link to download the appropriate log file and get

detailed information about the errors encountered during inventory upload. For detailed information on the **Inventory Upload Validation Errors** page, see the **Inventory Upload Validation Errors** page in the online help.

Appendix A- Oracle Tables and Views for Oracle Inventory Collection

You need to create a database user to collect Oracle inventory using any direct inventory collection methods. You should add the credentials of this user into the secured Password Store of all applicable inventory beacon. The database user must have read-only access to the following tables and views on each Oracle server present within the assigned subnet of an inventory beacon:

- `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`

Appendix B - Deploying FlexNet Inventory Agent on a Shared Location

In the agent-based inventory collection, all FlexNet inventory agents run according to the same inventory collection schedule. The portable nature of FlexNet inventory agent enables you to deploy it on a shared drive within your network in such a way that different devices can access it at different schedules. For example, a deployment of FlexNet inventory agent could be accessed by different Oracle servers to collect and upload discovery and inventory information to the inventory beacon at different times. While using a shared deployment, it is recommended to ensure that only one Oracle server accesses the shared deployment of the FlexNet inventory agent at a time. The total number of shared deployments depends on the network structure and access constraints. For example, if an Oracle server is not allowed to access the shared deployment of FlexNet inventory agent present in a different subnet, you may have to deploy another shared FlexNet inventory agent in the subnet of the target Oracle server.

When you deploy FlexNet inventory agent on a shared drive, you need to use the tools of your choice to schedule discovery and inventory collection for each Oracle server. You also need to ensure that the collected discovery

and inventory information (`.disco` and `.ndi` files) is transferred to the appropriate inventory beacon. The following are the steps to collect discovery and inventory information using this method:

For Windows

1. Copy the following files from the *InstallDir*`\RemoteExecution\Public\Inventory\` folder on an inventory beacon to a folder that is on or accessible to the target computer:

   - `ndtrack.exe`
   - `getSytemId.exe`
   - `mgscmn.dll`
   - `uploader.dll`
   - `wmitrack.ini`
   - `InventorySettings.xml.`

2. On the Oracle server, execute `ndtrack.exe` as a local SYSTEM user that is a member of the `ora_dba` group in the Oracle security settings. The `ndtrack` gathers the basic hardware and software inventory with Oracle inventory. The following are some example that describe the use of different `ndtrack` command line options. For more details about available `ndtrack` command line options, see the *Command Lines* chapter of the *FlexNet Inventory Agent and Managed Devices* guide available through the title page of the online help.

   - The command to gather inventory and upload it to a specific URL:

     ```
     ndtrack.exe -t Machine -o ShowIcon=false -o "LogFile=%TEMP%\ndtrack.log" -o Upload=true -o
       UploadLocation=http://your-beacon/ManageSoftRL
     ```

   - The command to gather inventory and save discovery and inventory files the `C:\Temp\Inventory` folder:

     ```
     ndtrack.exe -t Machine -o ShowIcon=false -o "LogFile=%TEMP%\ndtrack.log" -o Upload=false -
     o MachineZeroTouchDirectory=C:\Temp\Inventory
     ```

Note • In the case where you choose not to upload collected data directly to the inventory beacon, you can manually transfer the generated inventory (`.ndi`) files to the `Warehouse directory\Incoming\Inventories` folder and the discovery (`.disco`) files to the `Warehouse directory\Incoming\Discovery` folder on the inventory beacon. The inventory beacon uploads this information to FlexNet Manager Suite.

For UNIX

1. Copy the following files from the *InstallDir*`\RemoteExecution\Public\Inventory\` folder on an inventory beacon to a folder that is on or accessible to the target computer:

   - `ndtrack.sh`
   - `ndtrack.ini`

2. On the Oracle server, execute `ndtrack.sh` from an account with `ssh` privileges. This account must be registered this account in the secure Password Store on the appropriate inventory beacon. The `ndtrack` runs with sudo privileges and impersonates one of the accounts in the `dba` Oracle database group and gathers the basic hardware and software inventory with Oracle inventory. The following are some example that describe the use of different `ndtrack` command line options. For more details about available `ndtrack` command line options, see the *Command Lines* chapter of the *FlexNet Inventory Agent and Managed Devices* guide available through the title page of the online help.

- The command to gather discovery and inventory and upload it to a specific URL:

```
/bin/sh ndtrack.sh -o "LogFile=/var/tmp/ndtrack.log" -o Upload=true -o
 UploadLocation=http://your-beacon/ManageSoftRL
```

- The command to gather and save discovery and inventory files the `C:\Temp\Inventory` folder:

```
/bin/sh ndtrack.sh -o "LogFile=/var/tmp/ndtrack.log" -o Upload=false -o
 MachineZeroTouchDirectory=/var/tmp/Inventory
```

Note • In the case where you choose not to upload collected data directly to the inventory beacon, you can manually transfer the generated inventory (`.ndi`) files to the `Warehouse directory\Incoming\Inventories` folder and the discovery (`.disco`) files to the `Warehouse directory\Incoming\Discovery` folder on the inventory beacon. The inventory beacon uploads this information to FlexNet Manager Suite.

Appendix C - Inventory Collection when Inventory Beacon is Disconnected from FlexNet Agent

Flexera Software recommends you to setup automatic inventory collection through a direct connection between each Oracle server and the inventory beacon managing the subnet in which the Oracle server is present. If you have opted for (an inadvisable) disconnected mode (no connection between the FlexNet inventory agent and the inventory beacon), you have to perform the following actions:

- Copy the `InventorySettings.xml` file from each inventory beacon to the FlexNet inventory agent installation folder.

- Run the agent.

- From each FlexNet inventory agent, manually transfer the generated inventory (`.ndi`) files to the `Warehouse directory\Incoming\Inventories` folder and the discovery (`.disco`) files to the `Warehouse directory\Incoming\Discovery` folder on the inventory beacon. The inventory beacon uploads this information to FlexNet Manager Suite.

Note • *The FlexNet Inventory Agent and Managed Devices guide contains complete details about setting up the environment in the disconnected mode.*

Appendix D - Inventory Collection With Specific DBA

FlexNet Manager Suite enables you to run `ndtrack` (locally on an Oracle server through third-party scheduling tools or remotely from the appropriate inventory beacon) using a specific database administrator account. You can use this inventory collection method when you can not install FlexNet inventory agent on the target Oracle servers. This inventory collection method differs from the others because:

- It is only available for UNIX-based Oracle servers

- Without installing FlexNet inventory agent, this method collects Oracle database and application inventory with some hardware and other software inventory as well

- For this method, you do not need to know the `dba` group configuration (required for agent-based and zero touch inventory collection)

- As `ndtrack` runs as a specific database administrator account (and not as `root`), the collected hardware inventory may not be sufficient for the calculation of processor-based license calculations.

You can use this option in any of the two possible ways:

- **Locally on the target Oracle server**: You need to configure a schedule task (using third-party tools) to run `ndtrack` as the OS user that is a database administrator on the Oracle server. For testing purpose, you can also log in to the Oracle server using the specific database account and use the command line option to collect inventory.

- **Through zero touch inventory collection**: When using zero touch inventory collection (running from inventory beacon), you can modify the `ndtrack.ini` file to configure `ndtrack` to use the specified database account on the target Oracle server. You need to register the desired OS user that is a database admin on the Oracle server in the Password Store on the appropriate inventory beacon. The following are the steps to run `ndtrack` as a specific database administrator account:

1. Ensure that you have successfully deployed and configured inventory beacon(s).

2. When running locally on Oracle server, use third-party tools to create a schedule task to execute the following command as a specific database administrator account. The specified user is validated against the `dba` database group.

   ```
   ndtrack.sh -t machine -o UploadLocation=<URL>
   ```

   If you are executing `ndtrack` as root, you need to specify the database administrator account name. The specified user is not validated against the `dba` database user group. Consider the following example:

   ```
   ndtrack.sh -t machine -o OracleInventoryUser=<UserName> -o UploadLocation=<URL>
   ```

Note • *You can also use the `-o upload=false` option to stop uploads from the Oracle server to FlexNet Manager Suite. This option is generally helpful when troubleshooting inventory uploads.*

3. In case of remote execution of `ndtrack`, verify the operational status of the appropriate inventory beacon by checking the following inventory beacon properties on the **Discovery & Inventory** > **Beacons** page:

Property	Expected Value
Beacon status	`Operating normally`
Policy status	`Up to date`
Connectivity status	`Connected`

4. When using remote execution, ensure that you have created a user with `ssh` privileges and registered its credentials on the Password Store of the appropriate inventory beacon. The `ndtrack` will run with `sudo` privileges and use the specified database account to collect Oracle inventory. For more information, see the topics under **Password Management Page** in the online help.

5. (Optional) To specify a database administrator account as the default account for collecting Oracle inventory through `ndtrack`, perform the following activities:

`ndtrack` execution method	File name and location	Activity

Locally on the Oracle server	`config.ini` in the `/var/opt/managesoft/etc` directory	**1.** Add the following code:
		
		`[Managesoft/Tracker/CurrentVersion]`
		` OracleInventoryUser=oracledba`
		
		2. Run the following command:
		
		`ndtrack.sh -t machine -o UploadLocation=<URL>`

Remote execution from inventory beacon	`ndtrack.ini` in any of the following folders on the appropriate inventory beacon
	
	• `%ProgramFiles%\Flexera Software\Inventory Beacon\RemoteExecution\Public\Inventory`
	
	• `%ProgramFiles(x86)%\Flexera Software\Inventory Beacon\RemoteExecution\Public\Inventory`
		Add the following code:
		
		`[Managesoft/Tracker/CurrentVersion]`
		` OracleInventoryUser=oracledba`

4

FlexNet Inventory Scanner

Topics:

The FlexNet Inventory Scanner is a light-weight executable that you can run on any computer device to return inventory information for central analysis. It can save data locally (for manual inspection or for upload by other existing systems), or can upload collected inventory directly to your chosen location. A common configuration is to upload collected inventory to a inventory beacon, and from there it will be automatically forwarded to the central application server for FlexNet Manager Suite. It can conveniently be called from another tool, allowing powerful and configurable inventory collection without the overhead of deploying another agent.

A Windows executable is available for download. For non-Windows platforms, instructions to collect and set up the necessary files are included in this chapter.

The FlexNet Inventory Scanner, designed for the smallest possible system footprint, does not include any scheduling facilities. It can be run by your existing scheduling system, such as a Windows scheduled task, or Microsoft SCCM, a `cron` job on other platforms, or any other scheduling technology.

Its inventory collection functionality is identical to the FlexNet inventory agent that may also be deployed to computers, or perform remote (zero touch) inventory. The key advantage of FlexNet Inventory Scanner is its simplified deployment.

Comparison of Agent and Scanner

Because the full version of the FlexNet inventory agent and the lightweight FlexNet Inventory Scanner have such closely-related functionality, it is often challenging to keep separate sets of functionality clearly in mind. This table summarizes the differences between the two. This summary list may be helpful in finalizing a decision of which approach to use in inventory collection.

Keep in mind as well that the code objects must vary across platforms, so that many of the following details are different for Microsoft Windows platforms and UNIX-like platforms. Where the platforms are omitted in the table below, functionality is identical on all platforms.

A further level of complexity is added when we take account of zero touch inventory. In this case, the FlexNet Inventory Scanner is the code entity installed on the inventory beacon; but here its standard functionality is augmented by other code elements that are installed as part of FlexNet Beacon. These additional elements provide the zero touch capabilities (as described below), as well as managing the downloaded schedule and policy, and the necessary uploads. This combination of functionality from both the FlexNet Inventory Scanner and FlexNet Beacon implies a larger matrix, accommodating variations like this:

* The locally-installed, full FlexNet inventory agent includes scheduling, uploads, and usage tracking.

* Zero touch inventory includes scheduling, uploads, but no usage tracking,

* The stand-alone FlexNet Inventory Scanner does not include any scheduling, there are no uploads by default (although you can turn them on), and there is no usage tracking.

However, for simplicity, this table is limited to the two general cases: the full FlexNet inventory agent, and the FlexNet Inventory Scanner as deployed independently (not automatically). In the middle-ground case of zero touch inventory, details are included in the first column, since this is automated functionality built into FlexNet Manager Suite, and the combination of the FlexNet Inventory Scanner and FlexNet Beacon code provide functionality mapping closely to the complete FlexNet inventory agent (except for usage tracking).

Functionality	FlexNet inventory agent	FlexNet Inventory Scanner
Availability	Installed with FlexNet Manager Suite for automatic deployment. Additionally, downloadable through the web interface of FlexNet Manager Suite for those who want to manage deployment separately.	• Windows: Downloadable from the Product and License Center. • UNIX: Files must be collected from an installation of FlexNet Manager Suite, and packaged and deployed using third party utilities.
Configuration file for rollout (see also Preferences, below)	• Windows: `mgssetup.ini` • UNIX: `mgsft_rollout_response`	• Windows: `wmitrack.ini` • UNIX: `ndtrack.ini`

Functionality	FlexNet inventory agent	FlexNet Inventory Scanner
Deployment	Through 'adoption' (automated deployment by an inventory beacon to computers identified in a target that includes the adoption setting). *Tip • May also be deployed independently using existing tools and infrastructure; or manually.*	Third party deployment and installation using existing tools and infrastructure; or manual deployment.
Preferences storage	• Windows: Registry settings, `wmitrack.ini` • UNIX: `/var/opt/ managesoft/ etc/config.ini`	• Windows: None • UNIX: `ndtrack.ini`
Zero touch inventory	• Windows: A service created on the target device executes FlexNet Inventory Scanner from a network share, uploads the results, and is removed thereafter. • UNIX: Agent is installed (using `ssh`), executed, and then uninstalled.	• Windows: Not supported • UNIX: Not supported *Tip • Use of the FlexNet Inventory Scanner for collecting zero touch inventory is dependent on its default installation on an inventory beacon. Its use for zero touch inventory is not supported for independent deployment elsewhere.*

Downloading FlexNet Inventory Scanner for Windows

The fully self-contained executable is convenient to deploy.

To take inventory of computers running versions of Microsoft Windows, you can download the FlexNet Inventory Scanner, which is immediately ready to deploy.

1. Using the account details emailed to you with your order confirmation from Flexera Software, login to the Product and License Center at *https://flexerasoftware.flexnetoperations.com/*.

2. From the list of products, select <u>FlexNet Manager Platform</u>, and again in the following page used to differentiate elements of the Suite, if applicable.

3. Finally, click through the link for <u>FlexNet Manager Suite 2015 R2 SP3</u> to access the downloads.

4. Scroll down to the link for `FlexNet Inventory Scanner.zip`, and download it to a convenient folder (for example, `C:\temp`).

5. Expand (unzip) the downloaded archive, and do one of the following:

   • Manually transfer the executable file to the Windows computer where you plan to execute the FlexNet Inventory Scanner.

   • Save the executable to a network share drive that is accessible to several Windows computers from which you want to take inventory.

   • Package the executable for deployment and installation, using your preferred deployment tool.

     Tip • *A suitable working location might be* `C:\Program Files (x86)\Flexera Software \InventoryScanner.exe`*.*

6. If the FlexNet Inventory Scanner is required to discover Oracle listeners or take inventory of Oracle databases, also do the following:

   a) On an inventory beacon, navigate to `%ProgramFiles%\Flexera Software\Inventory Beacon \RemoteExecution\Public\Inventory`.

   b) Copy the file `InventorySettings.xml` (take note of the revision number in the first line of the file for future reference).

   c) Deploy this file, unchanged, into the same folder as the FlexNet Inventory Scanner is installed, using whichever of the above methods is appropriate.

   d) Set a reminder for yourself in your preferred tool to check the revision number of `InventorySettings.xml` after future updates of the Application Recognition Library. If the revision changes, you should manually update all the copies you manually deployed. These changes allow the inventory tool (whether the FlexNet Inventory Scanner, or the full FlexNet inventory agent that is kept up to date automatically) to stay abreast of inventory changes necessary to support the latest Oracle licenses.

The FlexNet Inventory Scanner is now ready to collect inventory on systems running Microsoft Windows (see the default behavior in *Default Operation (Windows)* on page 118).

Scanner for non-Windows Platforms

Although there is no ready-made download, you can still deploy a scanner for non-Windows devices.

The components needed for a light-weight scanner for UNIX-like platforms are readily available in all implementations of FlexNet Manager Suite. Using the following process to collect and prepare the components for deployment to a target inventory device.

Tip • *The required files are available only after the inventory beacon has downloaded its settings from the central application server.*

1. On an installed and registered inventory beacon, navigate to `BeaconInstallDir\RemoteExecution\Public\Inventory` (by default, this is `C:\Program Files (x86)\Flexera Software\Inventory Beacon\RemoteExecution\Public\Inventory`).

2. Collect a copy of each of the following files:

    • `ndtrack.sh` – this is the tracker (the core code element for both the lightweight FlexNet Inventory Scanner and the full, locally-installed FlexNet inventory agent).

    • `ndtrack.ini` – this is the configuration file for the tracker. This file is optional, and only needed if you wish to configure the behavior or settings for the tracker. If used, it must be installed in the same folder as the tracker. For more information about customizing the configuration, see *Configuring ndtrack.ini for UNIX-like Platforms* on page 125.

    • `InventorySettings.xml` – this file is unique for your implementation of FlexNet Manager Suite, and reflects the various options that you have licensed. The file is mandatory if you wish to collect Oracle inventory (or Microsoft Server inventory, but that is hardly relevant for UNIX-like platforms). If used, this file must also be installed in the same folder as the tracker. It may be omitted for a particular installation of the tracker where this installed instance does not collect Oracle inventory.

        Important • *Do not edit this file!*

3. Package these files, and deploy them either to target inventory devices, or to network shares accessible to your target devices.

    A typical location for the deployed files on a target device is under `/opt` or `/usr`, such as

    ```
    /usr/FlexeraSoftare/Scanner
    ```

4. Use `cron` or another scheduling tool of choice to invoke `ndtrack.sh`.

    Tip • *The command-line options for `ndtrack.sh` on UNIX-like platforms are identical to those for the FlexNet Inventory Scanner on Microsoft Windows. For details, see *FlexNet Inventory Scanner Command Line* on page 128.*

Default Operation (Windows)

This process assumes that you have downloaded the zip archive containing the FlexNet Inventory Scanner, and expanded the archive in a convenient location. Do one of the following:

- To run the FlexNet Inventory Scanner manually, navigate to the expanded archive in Windows Explorer and double-click `FlexeraInventoryScanner.exe`. Execution uses all the default values (see below).

- To run the FlexNet Inventory Scanner from the command line:

  1. Start a command window (for example, Windows **Start** button, type `cmd` in the search field, and press `Return`).

  2. In the command line window, navigate to the folder in the expanded archive where the executable is installed.

  3. Issue the command line, with any of the options documented in *FlexNet Inventory Scanner Command Line* on page 128.

- To run the FlexNet Inventory Scanner regularly, configure your preferred scheduling tool (such a Microsoft Task Scheduler) on the target device to regularly invoke the FlexNet Inventory Scanner, using the options available in *FlexNet Inventory Scanner Command Line* on page 128. The FlexNet Inventory Scanner may either be locally installed on this target device, or may be installed on a network share accessible from this target device.

At each invocation, the self-installing executable:

- Installs the Windows version of the tracker (or inventory collection) agent (`ndtrack.exe`) and related control files

- Executes `ndtrack` according to the command-line options supplied

- When execution is completed, uninstalls the `ndtrack` executable, leaving only the self-installing executable `FlexNet Inventory Scanner.exe` in place.

When no command-line options are specified, the tracker agent `ndtrack` does the following:

- Conducts a machine inventory on the local computer on which it is currently running.

- Saves this inventory in `%temp%\FlexeraSoftware\$(UserName) on $(MachineId).ndi`, where `%temp%` is the temporary directory for the account that is running the FlexNet Inventory Scanner, with the file name showing the account and machine ID related to the inventory run.

- Returns an exit code of `0` for success. For any other exit code, check the log file.

- Records the log for this activity in `%temp%\ManageSoft\tracker.log`.

- If `InventorySettings.xml` is supplied and `ndtrack` can connect to the Oracle database, returns a separate `.ndi` Oracle inventory file.

- If the Oracle listener can be found and queried, returns a discovery file with details of the Oracle listener.

Tip • `InventorySettings.xml` *can be copied from* `%ProgramFiles%\Flexera Software` `\Inventory Beacon\RemoteExecution\Public\Inventory` *on an inventory beacon, and dropped into the same folder as the FlexNet Inventory Scanner. Be aware that this file may be updated through the regular downloads of the Application Recognition Library, and automatically updated on the inventory beacons, so that your copy may subsequently need to be manually updated (check the revision number in the first line of the file).*

These behaviors can be changed with command-line options described in *FlexNet Inventory Scanner Command Line* on page 128. For example, you can instead configure the FlexNet Inventory Scanner to take a user-related inventory, and upload the resulting file immediately to an inventory beacon of your choice for integration into the standard inventory processes.

Default Operation (UNIX-like platforms)

You have deployed `ndtrack.sh`, optionally its customized configuration file `ndtrack.ini`, and the current and unedited `InventorySettings.xml` (required when Oracle inventory is in play). Do either of the following:

- To run `ndtrack.sh` from the command line, it's convenient to change directory to the location of these files (all in the same folder), and invoke the executable together with any further command-line parameters (documented in *FlexNet Inventory Scanner Command Line* on page 128).

- To run the lightweight scanner regularly, configure your preferred scheduling tool (such as `cron`) on the target device to regularly invoke `ndtrack.sh`, either using the options available in *FlexNet Inventory Scanner Command Line* on page 128 or using a customized copy of `ndtrack.ini` (see *Configuring ndtrack.ini for UNIX-like Platforms* on page 125). To execute the scanner, you can either use the `chmod` command to set the execute permissions on `ndtrack.sh`; or you can start `ndtrack.sh` using a shell (such as `/bin/sh /path/ndtrack.sh [options]`.

   Tip • *Experienced administrators can also combine use of* `ndtrack.ini` *(for common defaults) with matching command-line parameters (for local overrides on specific devices).*

You may execute `ndtrack.sh` either as `root`, or as a different user. However, only `root` allows collection of complete inventory. When `ndtrack.sh` executes as a non-root user, the following are amongst the inventory details that cannot be collected:

- File evidence from any file system path not accessible by the executing user

- InstallAnywhere, InstallShield Multiplatform, or Oracle Universal Installer evidence under paths not accessible by the executing user

- Oracle database service discovery via the local listener using `lsnrctl`

- Oracle database inventory which may use impersonation of an Oracle database administration user when running the `sqlplus` command

- On Linux systems:

- BIOS details (`dmidecode`): serial number, UUID, manufacturer, model, chassis type

- All hard disk information (from device files)

- On Solaris systems:

  - MAC addresses of network adapters

  - x86 BIOS details (`dmidecode`): model, manufacturer

  - SPARC model using OpenPROM interface (the tracker fails over to using the value from `sysinfo SI_PLATFORM` instead, which may give different results)

- On HP-UX systems:

  - SD-UX installation evidence from `swlist` if access has been locked down with `swreg` or `swacl`

  - vPar evidence including `VMType`, `VMName`, and vPar capacity (`vparstatus` requires root)

  - Hard disk drive properties including capacity

- On Mac OS X systems:

  - Mac OS X package bundle paths under `/Applications` or `/System/Library` not accessible by the executing user.

At each invocation, `ndtrack.sh`:

- Checks in its installed folder for a customized `ndtrack.ini` configuration file. If found, this file in its entirety replaces all the default values for operating preferences and data providers (see *Configuring ndtrack.ini for UNIX-like Platforms* on page 125).

- Uses any command-line parameters to override matching values (whether built-in defaults, or values from `ndtrack.ini`).

- Executes the inventory collection according to the resulting set of preferences.

When neither command-line options nor preferences in `ndtrack.ini` are specified, `ndtrack.sh` does the following:

- Conducts a machine inventory on the local computer on which it is currently running. (Only machine inventory is supported for UNIX-like platforms: there is no equivalent of the "user"-based inventory available on Microsoft Windows.)

- Creates an inventory file named `$(UserName) on $(MachineId).ndi`, where these variables are replaced as follows:

  - `$(UserName)` is replaced by the name of the account running the executable. When `ndtrack` is run as root, this value is returned as `system`, for consistency with inventory reported from Windows platforms.

  - `$(MachineId)` is replaced by the computer name of the inventory device, as returned from the operating system.

- Saves this inventory file (both uncompressed for inspection and compressed for upload) in either of the following paths:

  - When the executable has run as the `root` account, in `/var/tmp/flexera/uploads/inventories`

- When the executable has been run by any other user account (represented as *UserName*), in `/var/tmp/flexera.`*UserName*`/uploads/inventories`.

- Returns an exit code of `0` for success. For any other exit code, check the log file.

- Where `InventorySettings.xml` is supplied and `ndtrack.sh` can connect to database, creates a separate Oracle inventory file (such as `$(MachineId) at DateTimeInISO8601 (Oracle).ndi.gz`) in the same `inventories` folder.

- Where the Oracle listened is found and can be queried, creates a `.disco` discovery file containing details of the local Oracle database listener (such as `$(MachineId) at DateTimeInISO8601.disco`), saved in `/var/tmp/flexera/uploads/Discovery`.

- Records the log for these activities in `tracker.log` in either of the following locations:

  - When the executable has run as the `root` account, in `/var/tmp/flexera/log`

  - When the executable has been run by any other user account (represented as *UserName*), in `/var/tmp/flexera.`*UserName*`/log`.

These behaviors can be changed with command-line parameters described in *FlexNet Inventory Scanner Command Line* on page 128, or preferences saved in `ndtrack.ini` (see *Configuring ndtrack.ini for UNIX-like Platforms* on page 125). As a customization example, you can configure `ndtrack.sh` to upload the resulting inventory file immediately to an inventory beacon of your choice for integration into the standard inventory processes.

Configuring FlexNet Inventory Scanner

There are two ways to configure the inventory agent on inventory devices:

- At runtime, using command-line options (see *FlexNet Inventory Scanner Command Line* on page 128).

- Use the configuration file specific to the platform:

  - On Microsoft Windows, use the `wmitrack.ini` file that governs the Windows Management Instrumentation class tracking behavior. This is described in *Configuring WMI for FlexNet Inventory Scanner* on page 122.

  - On UNIX-like platforms, use the `ndtrack.ini` file that (like the `wmitrack.ini` file on Windows) controls the providers used for various kinds of inventory gathering; but unlike the other, can also be extended to include any of the preferences that may be set on the command-line.

Note • On Microsoft Windows, the FlexNet Inventory Scanner is different than the installed FlexNet inventory agent (where an inventory target has been 'adopted' by the local installation of the full inventory agent) in this regard:

- *The lightweight FlexNet Inventory Scanner does not access Windows registry settings to determine options.*

- *The full FlexNet inventory agent may access a wide range of registry settings to control its behavior.*

Configuring WMI for FlexNet Inventory Scanner

The behavior of FlexNet Inventory Scanner on Windows devices can (in part) be configured in a `wmitrack.ini` file.

The WMI (Windows Management Instrumentation) configuration file is used to inform the inventory agent what hardware, software and operating system components it should track. This file is only used if WMI tracking is selected as the preferred mechanism for hardware inventory tracking (set by the `WMI` preference, for which see *FlexNet Inventory Scanner Command Line* on page 128).

The components to be tracked can be any valid Win32 classes. A full list of supported classes is provided through the following URLs:

- Win32 classes: *http://msdn.microsoft.com/en-us/library/aa394084(v=vs.85).aspx*

- CIM classes: *http://msdn.microsoft.com/en-us/library/aa386179(v=vs.85).aspx*

- Software licensing classes: *http://msdn.microsoft.com/en-us/library/ee957720(v=vs.85).aspx*

You can edit this text-based file to change the items being tracked. For example, if you want to track user logons, edit the `wmitrack.ini` file to include the lines:

```
[Win32_ComputerSystem]
                UserName
```

The `UserName` value returns the name of the user currently logged on. In a terminal services session, `UserName` returns the name of the user that is logged on to the console — not the user logged on during the terminal service session. `UserName` may be null if the user currently logged on does not have administrative privileges. If you experience this problem, try using;

```
[Win32_LoggedOnUser]
                Antecedent
```

The user data collected will be available in hardware inventory reports.

You can also exclude an entire section, or an individual item, by commenting them out with a leading semi-colon. For example, in the sample file, see

```
;[Win32_USBDevice]
```

Locations

For remote inventory collection from Windows devices, the FlexNet Inventory Scanner checks the following in this order:

1. It looks for a command-line option `-o WMIConfigFile=` "*FullPathAndFileName*", and uses the file declared there (and no further checking occurs).

Note • There is no limitation on the file name or extension you may specify with this option.

2. In the absence of a command-line option, FlexNet Inventory Scanner looks for a `wmitrack.ini` file in the same folder where the self-executing zip expanded (`%temp%\FlexeraInventoryScanner`), on the computer where the scanner is executing. This is the standard operating location for this `ini` file.

If a `wmitrack.ini` file is not found in either way, no WMI hardware components are tracked.

Sample file

The following default `wmitrack.ini` file specifies the Win32 items collected during standard inventory tracking. (You may use this as source for modifying your own alternative configuration file.)

```
[Win32_ComputerSystem]
Manufacturer
Model
Domain
DomainRole
NumberOfProcessors
NumberOfLogicalProcessors
TotalPhysicalMemory
Status
UserName

[Win32_ComputerSystemProduct]
IdentifyingNumber
Name
UUID
Vendor
Version

[Win32_OperatingSystem]
Name
Manufacturer
Version
ServicePackMajorVersion
ServicePackMinorVersion
SerialNumber
InstallDate
LastBootUpTime
OSLanguage
FreePhysicalMemory
FreeVirtualMemory
CountryCode
WindowsDirectory
SystemDirectory
Caption
CSDVersion
Status
CSName
OSType
OSArchitecture

[Win32_BIOS]
Manufacturer
Version
ReleaseDate
SerialNumber
BiosCharacteristics
Status

[Win32_Processor]
Description
Manufacturer
Version
ProcessorId
CurrentClockSpeed
CurrentVoltage
L2CacheSize
Status
MaxClockSpeed
Name
ProcessorType
```

```
NumberOfLogicalProcessors
NumberOfCores
DeviceID

[Win32_DiskDrive]
Description
Manufacturer
Model
Size
InterfaceType
Partitions
Status

[Win32_LogicalDisk]
Description
VolumeName
FileSystem
FreeSpace
Size
VolumeSerialNumber
DriveType
MediaType
Status
ProviderName

[Win32_CDROMDrive]
Description
Manufacturer
Drive
Status
Capabilities

[Win32_NetworkAdapter]
Manufacturer
MACAddress
MaxSpeed
Speed
Status

[Win32_NetworkAdapterConfiguration]
Caption
Description
Index
MACAddress
IPEnabled
DHCPEnabled
IPAddress
DHCPServer
DNSHostName
DNSDomain
DNSServerSearchOrder
DefaultIPGateway
IPSubnet

[Win32_PhysicalMemory]
Capacity
MemoryType
PositionInRow
Speed
Status

[Win32_SoundDevice]
Name
Manufacturer

[Win32_VideoController]
Name
VideoProcessor
DriverVersion
DriverDate
InstalledDisplayDrivers
AdapterRAM

[Win32_VideoConfiguration]
AdapterRAM
AdapterType
```

```
Description
HorizontalResolution
MonitorManufacturer
MonitorType
Name
VerticalResolution

[Win32_SystemEnclosure]

;[Win32_USBDevice]
;Caption
;ClassGuid
;Description
;DeviceID
;Manufacturer
;Name
;Status
;SystemName

[SoftwareLicensingProduct]
ApplicationID
Description
EvaluationEndDate
GracePeriodRemaining
LicenseStatus
MachineURL
Name
OfflineInstallationId
PartialProductKey
ProcessorURL
ProductKeyID
ProductKeyURL
UseLicenseURL

[SoftwareLicensingService]
ClientMachineID
IsKeyManagementServiceMachine
KeyManagementServiceCurrentCount
KeyManagementServiceMachine
KeyManagementServiceProductKeyID
PolicyCacheRefreshRequired
RequiredClientCount
Version
VLActivationInterval
VLRenewalInterval
```

Configuring ndtrack.ini for UNIX-like Platforms

The optional configuration file ndtrack.ini can be used:

- To disable specific parts of inventory gathering (although doing this places at risk your ability to calculate consumption for licenses that rely on the inventory being available)

- To store run-time preferences that would otherwise be required on the command line at each invocation.

All the specifications embedded in the default ndtrack.ini file are also embedded in the ndtrack.sh executable, so that when the .ini file is omitted, default functionality is preserved. When the tracker is first invoked, it checks for the presence of an ndtrack.ini file in the same directory where the executable is running. If present, the external file takes precedence, *in its entirety*. This means that if, for example, your copy omits a provider statement that is present in the default file, this is equivalent to turning off that part of inventory gathering. For this reason, it is critically important to always start customization with a complete copy of the latest default file from your inventory beacon (see *Scanner for non-Windows Platforms* on page 117), and to change only those elements that are essential, preserving all other values.

Disabling a part of inventory

Find the appropriate section in the `ndtrack.ini` file, and remove or comment out the provider details. For example, the following change prevents collection of the physical memory size on Solaris platforms:

```
[ManageSoft\Tracker\CurrentVersion\SunOS\Hardware\MGS_PhysicalMemory]
#provider=SolarisPhysicalMemory
```

Note • A line commencing with the hash character (#) is commented out. This character is also known as the pound character, or number sign.

Be cautious about what you comment out, since an unusual variety of hardware attributes can affect consumption calculations on different kinds of licenses.

Storing a run-time preference

The behavior of the `ndtrack.sh` executable can be conditioned by a large number of preferences. When the parallel executable on Windows, `ndtrack.exe`, is operating within the locally-installed, complete FlexNet inventory agent, it can read these preferences either from the command line or from values saved in the Windows registry (in contrast, the Windows version of the light FlexNet Inventory Scanner does *not* check registry entries, simplifying its deployment and operation). On UNIX-like platforms, there is (of course) no Windows registry, so that preferences are either:

• Read from the command line as parameters, which parameters are common for both Windows and UNIX-like platforms (see *FlexNet Inventory Scanner Command Line* on page 128 for details)

• Saved in a configuration file that acts as an alternative storage medium on these non-Windows platforms.

On UNIX-like platforms, the configuration file is different in these two cases:

• When the complete FlexNet inventory agent is locally installed on the target device running a UNIX-like operating system, the file is called `config.ini`.

• When `ndtrack.sh` is being deployed alone as a light inventory scanner, the file is called `ndtrack.ini`.

The naming difference keeps clear their different purpose and differences in content; but these two files have one thing in common — their ability to store preferences for use by `ndtrack.sh` (acting as a 'virtual' registry). Furthermore, this common functionality is achieved in exactly the same way:

• The equivalent of the Windows registry key is listed inside square brackets

• The following lines under each key show the registry value (or values) set under that key.

For example, suppose that you want the collected inventory from your UNIX-like device uploaded to your inventory beacon. One way is to use the following two preferences on the command line (this example shows an IP address for the `UploadLocation`):

```
./ndtrack.sh -o Upload=True -o UploadLocation=http://198.51.100.3/ManageSoftRL
```

Another way is useful when you want to deploy the lightweight inventory scanner but by default have it regularly upload inventory. To achieve this, you can customize the `ndtrack.ini` file with the following addition,

and simply deploy this into the same directory as the executable `ndtrack.sh` (the `UploadLocation` can alternatively be the fully qualified server name):

```
[ManageSoft\Tracker\CurrentVersion]
Upload=True
UploadLocation=http://FQServerNameOrIP/ManageSoftRL
```

Such customizations require that you know the equivalent registry paths used on Microsoft Windows (as well as the name/value pairs). Many preferences, complete with the relevant registry paths, are documented in the *Preferences* chapter of *FlexNet Inventory Agent and Managed Devices* (`FlexNetInventoryAgentAndMDs.pdf`, available through the title page of online help).

For our previous example, that document shows the registry path for the computer-based preference `Upload` in this manner:

```
[Registry]\ManageSoft\Tracker\CurrentVersion
```

Here, the placeholder `[Registry]\` stands for one of the following values, as appropriate for the context:

* The registry base path on Windows 32-bit devices

* The registry base path on Windows 64-bit devices

* The `config.ini` file for the full FlexNet inventory agent locally installed on the device

* The `ndtrack.ini` file for the lightweight deployment of `ndtrack.sh` as a stand-alone inventory scanner.

In the case of the lightweight FlexNet Inventory Scanner, the placeholder `[Registry]\` should be read as "add the following path inside square brackets to your `ndtrack.ini` file". This reading, together with the information in the *Value/range* entry in the relevant topics, produces the example shown above, which is good for anonymous authentication on the upload. The standard URL construct can also be used where an account name and password are required for the upload, although you may prefer this format on a transitory command line rather than in a plain text file:

```
[ManageSoft\Tracker\CurrentVersion]
Upload=True
UploadLocation=http://AccountName:Password@FQServerNameOrIP/ManageSoftRL
```

In a similar manner, you can include in your `ndtrack.ini` file any other preferences for `ndtrack` from `FlexNetInventoryAgentAndMDs.pdf` that are relevant to UNIX-like platforms.

Note • *Any preference setting in* `ndtrack.ini` *is over-ridden by a different value for the same preference given as a command-line parameter. The priority order is: default (built in) values are over-ridden by settings in* `ndtrack.ini`, *which are over-ridden by command-line parameters.*

Customizing Searches for FlexNet Inventory Scanner

You can extensively customize the FlexNet Inventory Scanner behavior using the command line by instructing it to include or exclude any of the following:

- Folders (and optionally, their subfolders)

- Particular file extensions

- Specific filenames

- Specific MD5 digest values.

Tip • *Checking MD5 digests across all inventory can significantly extend the time required for gathering inventory on a device.*

When including or excluding extensions, file names and MD5 values, there is a possibility that a file could be both included and excluded. To overcome this problem, a matching MD5 value overrides a file name, which overrides a file extension.

For example, if

- File extension `exe` is included

- Filename `xcopy.exe` is excluded

- MD5 value `123456...` (the MD5 for `xcopy.exe`) is included

then the FlexNet Inventory Scanner includes all files with extension `exe` except for all versions of `xcopy.exe` that do not have an MD5 value `123456...`

If the same directory, file extension, filename or MD5 value is explicitly included and excluded, the exclusion command takes precedence.

All such customizations are available through command-line options when you execute the FlexNet Inventory Scanner (see *FlexNet Inventory Scanner Command Line* on page 128). On UNIX-like platforms only, in addition to the command-line options, they may be configured in the `ndtrack.ini` configuration file (see *Configuring ndtrack.ini for UNIX-like Platforms* on page 125).

FlexNet Inventory Scanner Command Line

Command line reference.
(On Microsoft Windows) `FlexeraInventoryScanner.exe [options...]`
(On UNIX-like platforms) `ndtrack.sh [options...]`

Options:

`-o` *tag* = *value*

These parameters override the default FlexNet Inventory Scanner settings on Windows. On UNIX-like platforms, they override both the default settings and any preferences recorded in `ndtrack.ini`.

Possible tags are listed in the following table (and see also the notes following the table). Enclose values in double quotation marks if the values include spaces; otherwise, double quotation marks are optional. Special characters (double quotation marks, backslash) must be escaped with a backslash.

Tip • The `-t` command-line option available on Windows for the installed FlexNet inventory agent is not supported for the FlexNet Inventory Scanner. Instead, you can specify the inventory type using the command line parameter:

```
-o InventoryType=User|Machine
```

If the `InventoryType` preference is not specified, the type of inventory collected depends on the account running the FlexNet Inventory Scanner:

- *For the **LocalSystem** account, a computer (machine) inventory is collected*

- *For all other accounts, a user inventory is collected.*

For UNIX-like platforms, only machine-based inventory is supported.

Return codes

FlexNet Inventory Scanner returns a zero on success. If you receive a non-zero return code, check the log file. Details of the log file may also be configured with command-line options, as listed below.

Command line examples

This example collects a computer inventory and stores it locally (on the computer device where the FlexNet Inventory Scanner is executing) for upload by a separate system:

```
FlexeraInventoryScanner.exe
        -o InventoryType=Machine
        -o MachineZeroTouchDirectory="local-folder"
        -o Upload=False
```

This example collects user-based inventory and uploads it to an inventory beacon:

```
FlexeraInventoryScanner.exe
        -o InventoryType=User
        -o UploadLocation="http://InventoryBeacon/ManageSoftRL"
```

Options

Except as specifically noted, the same options are supported for the FlexNet Inventory Scanner on Microsoft Windows, and for the use of `ndtrack.sh` as a scanner on UNIX-like platforms.

Option tag	Default value	Notes
DateTimeFormat	%Y%m%dT%H%M%S	Based on the ANSI C function `strftime()`, sets the date/time format for all FlexNet Inventory Scanner activity. Since the

Option tag	Default value	Notes
		`DateTimeFormat` string is also used as a file name, it may contain only characters that are valid in file names across all platforms (in particular, you may not use a colon). *Warning • Internal use only: do not edit.*
`Domain`	(No default: may be any valid domain name.)	Contains the name of the domain of the local computing device on which the FlexNet Inventory Scanner is executing. If no value is set, the current local domain is used on Microsoft Windows platforms. On UNIX-like platforms, this preference may be used to set a domain name within which the inventory devices are to be reported.
`ExcludeDirectory`	(No default.)	Excludes a specified folder from inventory. If `Recurse` is `True`, then all subfolders are also excluded. This preference can accept multiple values (see notes below). If a folder is identified in both the `ExcludeDirectory` and `IncludeDirectory` preferences, it is excluded. Exclusions always override inclusions. Example: `$(WinDirectory)`
`ExcludeExtension`	(No default.)	For files within a folder included in inventory, FlexNet Inventory Scanner excludes files with the specified extension from inventory, or excludes all files if set to the value `*` (asterisk) . This preference can accept multiple values. You can specify any valid file extensions (no leading dot required). Be aware that including `exe` in this list prevents tracking of executable files on Windows platforms.
`ExcludeFile`	(No default.)	For files within a folder included in inventory, FlexNet Inventory Scanner excludes a specific file from inventory collection. This preference can accept multiple values.
`ExcludeMD5`	(No default.)	For files within a folder included in inventory, FlexNet Inventory Scanner performs an MD5 checksum, and excludes any files from the

Option tag	Default value	Notes
		inventory that have an MD5 value equal to any value stored in this preference. This preference can accept multiple values. Any valid MD5 can be specified: for example, `7d9d2440656fdb3645f6734465678c60`
GenerateMD5	False	Boolean (`True` or `False`). Specifies whether or not to calculate the MD5 digest of any file being tracked by the inventory agent and include it with stored inventory data. Since MD5 digests are not used to identify versions of inventoried files, set this preference to `True` only when it is needed to support an `IncludeMD5` or `ExcludeMD5` preference.
Hardware	True	Boolean (`True` or `False`). Allows you to track hardware either using Windows Management Instrumentation (WMI) or native APIs. If WMI is available, it is used for tracking. This preference is only effective when running in the machine context. (To track hardware in the user context, use `UserHardware`.) When set to `True` , allows the tracking of hardware inventory. When set to `False` , does not track hardware inventory.
IncludeDirectory	(Blank)	Includes a specified folder for collecting inventory. If `Recurse` is `True`, then all subfolders are also included. When the value of this entry is set to "\", it means "include all folders on all local drives". (For UNIX-like platforms, use forward slashes; and "/" includes the entire file system on all connected hard drivesS.) This preference can accept multiple values, separated by semi-colons (see note below). If a folder is identified in both the `ExcludeDirectory` and `IncludeDirectory` preferences, it is excluded. Exclusions always override inclusions. 🔲 *Important • Since the default is blank, this means that no files are included in inventory*

Option tag	Default value	Notes
		gathering by default. Inventory then relies entirely on installer evidence. If you wish to collect file evidence for any reason, it is mandatory to set an appropriate value for `IncludeDirectory`*.*
`IncludeExecutables`	`True`	Boolean (`True` or `False`). If `IncludeDirectory` is blank (its default), this setting has no effect. However, when `IncludeDirectory` has any value: • A setting of `False` means that only files with a file extension of `.exe` are included in executable files inventory (that is, executables on UNIX-like platforms are excluded). • A setting of `True` means that (for files not already matched by other settings) the FlexNet Inventory Scanner will report: • On Windows, files with an `exe` filename extension • On UNIX-like platforms, files with no filename extension and an execute bit set (in any of local, group, or universal scope in the file permission bits).
`IncludeExtension`	`sys;sys2;swtag;cmptag;lax;exe;`	For files within a folder included in inventory, inventory includes files with the specified extension, or includes all files if this preference is set to the value `*`. This preference can accept multiple values, separated with semi-colons. Example: `-o IncludeExtension=bat`
`IncludeFile`	(No default.)	The FlexNet Inventory Scanner searches for the specific file; but keep in mind that the search is constrained to folders that are included in inventory. This preference can accept multiple values, separated with semi-colons. Example: `-o IncludeFile=myfile.txt`
`Include MachineInventory`	`True` if running as **LocalSystem** or running a machine inventory on the	Boolean (`True` or `False`). If `True`, FlexNet Inventory Scanner performs a computer inventory including hardware and all user packages.

Option tag	Default value	Notes
	command line. `True` for UNIX-like platforms.	
`IncludeMD5`	(No default.)	For files within a folder included in inventory, FlexNet Inventory Scanner includes any having the specific MD5 digest. This preference can accept multiple values, separated with semi-colons.
`IncludeRegistryKey`	If no value is specified, FlexNet Inventory Scanner uses `HKEY_LOCAL_MACHINE\SOFTWARE\ Microsoft\Windows \CurrentVersion \Uninstall\`	Set this preference to instruct the FlexNet Inventory Scanner to return the contents of the specified registry keys or values on Microsoft Windows platforms. In order to collect all values under a specified key, the key path specified must end with a trailing backslash. If the path specified corresponds to a key (rather than a registry value) but does not end with a trailing backslash, only the `(Default)` value (if it is set) for the specified key will be collected. *Note • Default values in the registry are typically not set.* For example: • `HKLM\SOFTWARE\ManageSoft Corp \ManageSoft\` will track all values under the specified key (because it has a trailing backslash) • `HKLM\SOFTWARE\ManageSoft Corp \ManageSoft` will only track the `(Default)` values under the specified key (where they exist). When setting this preference, you can use: • The `*` wildcard to replace a key or value (including multiple times for different key elements in a single path) • The abbreviations `HKLM`, `HKCU`, `HKCR`, `HKU`, `HKCC`. These will be automatically expanded to appropriate values. This preference is ignored for UNIX-like platforms.

Option tag	Default value	Notes
Include UserInventory	True if running as user or running a user inventory (-o InventoryType=User on the command line).	Boolean (True or False). If True, perform a user inventory on Microsoft Windows platforms (there is no user inventory available on UNIX-like platforms).
InventoryFile	$(UserName) on $(MachineId).ndi	Identifies the name of a local copy of the inventory file. The name may consist of Windows properties that can be expanded to identify a value. For example, the default value $(UserName) on $(MachineId).ndi expands so that the name contains the account and machine ID related to the inventory run.
Inventory ScriptsDir	$(CommonAppDataFolder) \ManageSoft Corp \ManageSoft	The location of scripts to be run immediately before inventory data is uploaded through the inventory beacon. All scripts that exist in this location are run.
InventoryType	User unless the account executing FlexNet Inventory Scanner is LocalSystem, in which case the default switches to Machine.	Identifies the inventory type, either machine-based or user-based, for Microsoft Windows platforms. On UNIX-like platforms, only machine-based inventory is supported, and this setting is ignored.
LogFile	$(TempDirectory)\ManageSoft \tracker.log	The location of the log file (on the computer device where the scanner is executing) when logging is enabled for the FlexNet Inventory Scanner.
LogLevel	A-z (all logging enabled).	The level of logging used by the FlexNet Inventory Scanner. This logging is output to the file whose name is stored in the LogFile entry. Individual entries include the following: • A - Schedule logging • C - Callout logging • G - General logging • N - Network logging • P - Preference logging • S - Security logging • U - User interface logging

Option tag	Default value	Notes
		• `V` - Verification logging
`LogModules`	On Windows: `default;C:\Program Files\ManageSoft\Common\mgssyslg.dll` On Unix-like systems: `default;$(InstallDir)/lib/levtlog.so`	The modules used to log events for FlexNet Inventory Scanner. When unspecified, the default inventory agent log modules are used.
`LowProfile`	Default behavior is `True`	Determines the CPU priority of FlexNet Inventory Scanner on the computer device where it is executing. • When set to `True`, FlexNet Inventory Scanner processes run with low priority. • When set to `False`, the same processes run with normal priority.
`MachineName`	(No default.) If no value is set, FlexNet Inventory Scanner uses the current machine name.	Contains the name of the local machine. Unlike `MachineId`, this preference should not be changed. *Note • For UNIX-like platforms, `MachineId` can be set on the command-line, or in the `ndtrack.ini` configuration file. For details, see the preferences documented in the FlexNet Inventory Agent and Managed Devices PDF file, available through the title page of online help.*
`Machine ZeroTouch Directory`	`%temp%\FlexeraSoftware\`, which is in the temporary directory for the account running the inventory scan.	Microsoft Windows directory where machine inventory files are written (temporarily, pending upload) during any inventory gathering by FlexNet Inventory Scanner. If the upload (called as part of the inventory scanning process) proceeds normally, each temporary file is cleaned up after upload. *Note • The FlexNet Inventory Scanner uses this setting for any machine inventory, even though*

Option tag	Default value	Notes
		it is gathering inventory locally on the same computer. This preference is ignored on UNIX-like platforms.
MSI	True	Boolean (`True` or `False`). When set to `True`, Microsoft Installer (MSI) package information is added to the inventories. When set to `False`, FlexNet Inventory Scanner does not include MSI package information in inventories. This preference is ignored on UNIX-like platforms.

`ProgramFiles`
`ProgramFilesX86Folder`
`ProgramFilesX64Folder`
	Defaults vary by version of Windows.	A set of options that point to the Windows program files folder (across various versions of the operating system) on the target device where inventory is being gathered. `Program Files` exists for backwards compatibility; current distributions operate using `ProgramFilesX86Folder`.
		
		Note • `ndtrack` (the executable underlying the FlexNet Inventory Scanner) is a 32-bit executable.
		
		Default values respectively on a 32-bit platform are:
		
		• `Program Files`
		
		• `Program Files`
		
		• "" (empty value).
		
		Default values on a 64-bit platform are:
		
		• `Program Files`
		
		• `Program Files (x86)`
		
		• `Program Files`
		
		This preference is ignored on UNIX-like platforms.

Recurse	True	Boolean (`True` or `False`). When set to `True`, FlexNet Inventory Scanner includes folders

Option tag	Default value	Notes
		beneath the top-level folder specified by `IncludeDirectory`. When set to `False`, FlexNet Inventory Scanner does not recurse folders beneath the top level folder. It only tracks files immediately within included folder(s).
Run InventoryScripts	False	Boolean (`True` or `False`). When `True`, this preference specifies that inventory scripts should be run after Windows-based managed devices have been inventoried. All scripts located in the location specified by `InventoryScriptsDir` are then executed immediately after inventory data collection is complete. This preference is ignored for UNIX-like platforms.
ShowIcon	Default behavior is `True`	Boolean (`True` or `False`). When set to `True`, FlexNet Inventory Scanner displays an icon in the system tray when it is collecting inventory. This icon displays, regardless of the value of the `UserInteractionLevel` preference. If this icon is double-clicked and `UserInteractionLevel` is set to `Status` or `Auto`, the progress display toggles from being hidden to being visible. When set to `False`, no icon will display. This preference is ignored for UNIX-like platforms.
SysDirectory	Value	`SysDirectory` is a FlexNet pre-defined variable for the path to the Windows `System` folder. There is also a corresponding `SystemFolder` Windows pre-defined variable for the same folder, which you may also reference. This preference is ignored for UNIX-like platforms.
Upload	False	Boolean (`True` or `False`). When set to `True`, the inventory files generated by FlexNet Inventory Scanner are immediately uploaded to the reporting location (see `UploadLocation`).

Option tag	Default value	Notes
		When set to `False`, the inventory files are not uploaded (for example, in case you wish to inspect or validate their contents). 💡 *Tip • The default value for FlexNet Inventory Scanner is the inverse of the default for the installed, complete FlexNet inventory agent.*

UploadLocation	(No default.)	Mandatory when the `Upload` parameter is `True`, and otherwise ignored. The value must be a URI of the form:
		
		 ```protocol separator server-name/location```
		
		 where
		
		 • *protocol* and *separator* comprise one of these combinations (note three slashes for the file protocol):
		
		   • `http://`
		   • `https://`
		   • `ftp://`
		   • `file:///`
		
		 💡
		 Tip • If you are uploading to an inventory beacon, use one of the HTTP or HTTPS protocols. The FlexNet Beacon software does not provide native support for FTP or file shares. (In addition, on UNIX-like platforms, network shares are not supported.)
		
		 • *server-name* is either the IP address or a server name (where the system running the FlexNet Inventory Scanner has DNS access to resolve server names) of the destination for the upload (such as a parent inventory beacon, or a central application server).
		
		 💡

Option tag	Default value	Notes
		Tip • *If the target server is not listening on the default port number for the selected protocol, add the port number to the server name, separated by a colon. Example:* `https://myServer.example.com:886/ManageSoftRL` • *location* is the reporting location that receives uploads on the destination server. In a default installation, this may be called `ManageSoftRL`, but may have been given any name during installation. This is a web service that receives the uploaded inventory, and stores it (briefly) in `%CommonAppData%\Flexera Software\Incoming\Inventories` on the inventory beacon. The scheduled task `Upload FlexNet logs and inventories`, which by default runs every minute, then transfers the file to the central server.
UserHardware	False	Boolean (`True` or `False`). Allows you to track hardware either using Windows Management Instrumentation (WMI) or native APIs. If WMI is available, it is used for tracking. This preference is only effective when running in the user context. To track hardware in the machine context, use `Hardware`. When set to `True`, allows the tracking of hardware inventory. When set to `False`, does not track hardware inventory. On UNIX-like platforms, where only machine inventory is supported, this setting is ignored.
UserInteraction Level	Status	The user interaction method of the FlexNet Inventory Scanner. Possible values are: • `Full`: FlexNet Inventory Scanner operates in full interactive mode. • `Auto`: When `ShowIcon` is `True`, the FlexNet Inventory Scanner icon displays during inventory activities. The user is able to double-click the icon to access the FlexNet

Option tag	Default value	Notes
		Inventory Scanner user interface. When `ShowIcon` is `False`, a progress bar displays during inventory activities. (Since `ShowIcon` is not supported on UNIX-like platforms, `Auto` is not useful there.) • `Quiet`: FlexNet Inventory Scanner is not displayed during operations, and no user feedback or interaction is available. • `Status`: Only status dialogs are displayed (for example, progress dialogs).
UserZeroTouch Directory	(No default.)	Directory where user inventory files are written (temporarily, pending upload) during a remote ("zero touch") inventory gathering process. If the upload (called as part of the inventory scanning process) proceeds normally, each temporary file is cleaned up after upload. *Note • The FlexNet Inventory Scanner uses this option for any user-based inventory, despite the fact that it is collecting inventory locally on the same computer. (Also note that the default is different than the default for the installed inventory agent.)* On UNIX-like platforms, where only machine inventory is supported, this setting is ignored.
VersionInfo	True	Boolean (`True` or `False`). When set to `True`, FlexNet Inventory Scanner includes Windows file version header information in the inventory. When set to `False`, FlexNet Inventory Scanner does not include file version header information in the inventory. This preference is ignored for UNIX-like platforms.
WinDirectory	C:\Windows	The path to the Windows folder. You can also use `WindowsFolder`: this is a Windows pre-defined variable for the same folder. This preference is ignored for UNIX-like platforms.

Option tag	Default value	Notes
WMI	When taking machine-based inventory: `True`. When taking user-based inventory: `False`	Boolean (`True` or `False`). When set to `True`, the Windows Management Instrumentation (WMI) tracking is specified as the preferred option for tracking hardware. In this case, if WMI is not available (and the `Hardware` preference is set to `True`), FlexNet Inventory Scanner attempts to track hardware using a native API. When set to `False`, FlexNet Inventory Scanner uses another tracking mechanism instead of WMI. This preference is ignored for UNIX-like platforms.
WMIConfigFile.	`$(ProgramPath)\ wmitrack.ini`	The location of the Windows Management Instrumentation (WMI) configuration file, used to inform the FlexNet Inventory Scanner what hardware components it should track. This is only used if `WMI` is `True`. This preference is ignored for UNIX-like platforms.

Notes

1. Default values only apply if the parameter is not specified.

2. Directory paths must be specified as absolute paths (that is, on Windows, starting with a drive name). The typical wildcards in directory names are supported (`*` representing any number of characters, and `?` representing a single character).

3. The FlexNet Inventory Scanner accepts all name/value combinations, although if a preference is used that does not appear in the list above, it may be ignored. On UNIX-like platforms, other preferences may be included in the `ndtrack.ini` configuration file (not supported for Microsoft Windows).

4. A preference value can symbolically refer to another preference by enclosing its name thus: `$(preferenceName)`. References can contain further references.

   Example: The command

   ```
   FlexeraInventoryScanner.exe -o IncludeDirectory=$(WinDirectory)
   ```

   includes the Windows directory in the scan. References are resolved after all preferences are loaded so there are no ordering issues. Hopefully it is self-evident that, on UNIX-like platforms, only supported preferences can be referenced.

5. Semicolon or comma-separated values are the only method for defining multiple values in the FlexNet Inventory Scanner. Only the `Include` and `Exclude` preferences listed above may have multiple values. All other preferences contain a single value that can be overwritten.

Example: The command

```
FlexeraInventoryScanner.exe -o IncludeDirectory=C:\\;D:\\;E:\\
```

will scan the computer's C:, D:, and E: drives if they are fixed (hard) disks, but not if they are CD-ROM drives or logical drives (mapped to network locations).

5

Server Scheduling

Topics:

- *Server-Side Scheduled Tasks*
- *Introducing the Batch Scheduler*

FlexNet Manager Suite relies on a significant numbers of schedules. For example, there are schedules on the inventory beacons that determine when inventory data is collected from third-party inventory systems, from Active Directory, and so on. There is a schedule, set centrally, that determines when installed FlexNet inventory agents collect data from their hosts, and upload resulting discovery and inventory files to an inventory beacon. Both of these kinds of schedule operate remotely, away from the central application server.

However, there are also a significant number of scheduled tasks that run on the central application server. These schedules that operate on the application server are covered in this chapter.

Traditionally, scheduled tasks have been managed by the Microsoft Task Scheduler on the server. To maintain accessibility, these forms of task scheduling are still available, and are summarized in this chapter. Many of these Microsoft scheduled tasks simply queue messages for the internal FlexNet batch scheduler that runs on the application server. (In larger implementations where there are multiple servers, the batch scheduler runs on the server known as the batch server.)

The FlexNet batch scheduler incorporates knowledge of the interdependencies between the processes used in FlexNet Manager Suite. It is therefore able to make optimal scheduling decisions that avoid system conflicts and go some way towards load balancing on the batch server. The batch scheduler and batch processor are also covered in this chapter.

Server-Side Scheduled Tasks

In general, the scheduled tasks that are (by default) set up in Microsoft Task Scheduler on your central application server fall into three groups:

- Calls to `mgsimport.exe` that collect various kinds of data from staging locations on the inventory server and import into the *inventory* database. Examples include Active Directory, FlexNet inventory, installation logs, usage records, and VDI information. In a multi-server implementation, these scheduled tasks execute on the inventory server. Data handled this way requires an additional import from the inventory database to the compliance database (executed by the batch processor, as described shortly).

- Calls to the same `mgsimport.exe` that collect different kinds of data from staging locations on the inventory server and import directly into the compliance database. Examples include inventory beacon state and activity status, and discovery information. In a multi-server implementation, these scheduled tasks execute on the inventory server. Clearly, since these kinds of data have been loaded directly into the compliance database, there is no further import required.

- Calls to the internal batch scheduler to queue various tasks, taking into account all the system interdependencies and constraints. Examples include download of the product libraries, imports from the inventory database to the compliance database, and others shown below. In a multi-server implementation, these scheduled tasks execute on the batch server. For more information on the resulting batch scheduler tasks, see *Batch Scheduler Command Line* on page 156.

The default configuration of the Microsoft scheduled tasks is shown in the table below.

Scheduled task	Default schedule	Command	Note
Data warehouse export	6am Sunday	`BatchProcessTask.exe run DataWarehouseUpdate`	Export data as a snapshot to the data warehouse database (on a multi-tenant system, this is for all tenants).
Delete activity log history	4am daily	`BatchProcessTask.exe run ActivityLogHistoryDelete`	Truncate the execution history of scheduled batch processes.
Export to ServiceNow	3am Sunday	`BatchProcessTask.exe run ServiceNowExport`	When FlexNet Manager Suite integrates with ServiceNow, ServiceNow is regarded as authoritative for asset and contract records. This process synchronizes this information. For details, see the chapter *ServiceNow Integration with FlexNet Manager Suite* in *FlexNet Manager Suite Adapters Reference*, available through the title page of online help.
FlexNet inventory data maintenance	Midnight daily	`BatchProcessTask.exe run IMDataMaintenance`	Performs cleanup on the inventory database (containing FlexNet inventory) in FlexNet Manager

Scheduled task	Default schedule	Command	Note
			Suite. This also adds discovery information for systems found in inventory but not previously discovered.
FlexNet Manager Suite database support task	Midnight daily	`BatchProcessTask.exe run FNMPDataMaintenance`	Performs cleanup on the compliance database in FlexNet Manager Suite.
FNMEA Enterprise Groups export	1am daily	`BatchProcessTask.exe run FNMEAEnterpriseGroupExport`	Updates FlexNet Manager for Engineering Applications with the latest structure of enterprise groups recorded in FlexNet Manager Suite.
Import Active Directory	Every 10 minutes	`mgsimport.exe -o CREATE_NO_WINDOW=True -- -t activedirectory`	Imports into the inventory database any Active Directory data that has been uploaded from the inventory beacons to the `Incoming` folder. The availability of these files depends on Active Directory imports being scheduled on at least one inventory beacon. When this import is completed, a message is queued for the batch scheduler to execute `BatchProcessTask.exe run ActiveDirectoryImport --"-s` *connectionName* `-e ActiveDirectory"`, where *connectionName* is replaced with the connection to the inventory database. In a multi-tenant system, the batch processing is specific to each tenant.
Import application usage logs	Every 10 minutes	`mgsimport.exe -o CREATE_NO_WINDOW=True -- -t usagedata`	When usage tracking is configured for FlexNet inventory agent, this command imports uploaded usage records from the `Incoming` folder to the inventory database. This data is collected in the next inventory import (worst case is typically the daily catchup task).

Scheduled task	Default schedule	Command	Note
Import discovery information	Every 10 minutes	`mgsimport.exe -o CREATE_NO_WINDOW=True -- -t discovery`	Imports uploaded discovery records from the `Incoming` folder directly to the compliance database.
Import installation logs	Every 10 minutes	`mgsimport.exe -o CREATE_NO_WINDOW=True -- -t logs`	Loads into the inventory database the installation records of packages required by the FlexNet inventory agent, covering schedule changes, updated rules and so on. This data is resolved into the compliance database along with the next inventory import, where currently it is use as installation evidence for the FlexNet inventory agent on the managed device.
Import inventories	Every 10 minutes	`mgsimport.exe -o CREATE_NO_WINDOW=True -- -t inventories`	Imports all the inventory data collected by the FlexNet inventory agent and uploaded through inventory beacons, loading it from the `Incoming` folder to the inventory database. This data is collected in the next inventory import (worst case is typically the daily catchup task).
Import Inventory Beacon activity status	Every 10 minutes	`mgsimport.exe -o CREATE_NO_WINDOW=True -- -t activitystatus`	Loads the activity (or process) reporting of all available inventory beacons directly to the compliance database. Examples include rule execution and gathering inventory from third-party connections (such as Microsoft SCCM). This data feeds the listing available in the system menu (⚙▼ in the top right corner) > **Data Inputs**.
Import Inventory Beacon status	Every 10 minutes	`mgsimport.exe -o CREATE_NO_WINDOW=True -- -t beaconstatus`	Loads the state (or condition) reporting of all available inventory beacons directly to the compliance database. Examples include whether the FlexNet Beacon software has any known issues, the state of its services, its settings for uploading to its parent (either the

Scheduled task	Default schedule	Command	Note
			central application server or another inventory beacon in the hierarchy), and the like. This data feeds the listing available in **Discovery & Inventory** >**Beacons** (in the **Network** group).
Import remote task status information	Every 10 minutes	`mgsimport.exe -o CREATE_NO_WINDOW=True -- -t ActionStatus`	Deprecated: no remaining function. You may disable this task.
Import SAP inventories	10pm daily	`BatchProcessTask.exe run SAPInventoryImport`	Imports inventory from FlexNet Manager for SAP Applications that has been uploaded into the `Incoming` folder, writing the results into the compliance database.
Import SAP package license	6am Sunday	`BatchProcessTask.exe run SAPPackageLicenseImport`	Imports the licenses calculated by SAP, writing results into the compliance database.
Import SAP user and activity information	Midnight Sunday	`BatchProcessTask.exe run SAPUserAndActivityImport`	Perform a SAP user and activity information import.
Import security event information	Every 10 minutes	`mgsimport.exe -o CREATE_NO_WINDOW=True -- -t securityevent`	Deprecated: no remaining function. You may disable this task.
Import system status information	Every 10 minutes	`mgsimport.exe -o CREATE_NO_WINDOW=True -- -t systemstatus`	Deprecated: no remaining function. You may disable this task.
Import VDI access data	Every 10 minutes	`mgsimport.exe -o CREATE_NO_WINDOW=True -- -t vdiAccess`	Imports data on which users can access virtual desktops from the `Incoming` folder to the inventory database. This data is collected in the next inventory import (worst case is typically the daily catchup task).
Inventory import and license reconcile	2am daily	`BatchProcessTask.exe run InventoryImport`	Perform a full inventory import from all connections, followed by a license recalculation (reconciliation). In a multi-tenant implementation, this is for all tenants. For more information, see

Scheduled task	Default schedule	Command	Note
			the corresponding entry in *Batch Scheduler Command Line* on page 156.
Recognition data import	1am daily	`BatchProcessTask.exe run ARLDownload`	Schedules the download of the Application Recognition Library to the `%TEMP%` folder of the service account running the download. Only one instance of the ARL download task can run at a time. The batch scheduler automatically adds follow-on tasks for the import and clean up of the downloaded data.
Regenerate Business Import config	5:30am daily	`BatchProcessTask.exe run BusinessAdapterConfig`	Updates the data model (`FNMPDataModel.ini`), the DDI files, and the CSV templates used by the Business Importer (and therefore the Business Adapter Studio), and initiates downloads to the inventory beacons. This update is particularly valuable for keeping the Business Importer up to date with changes to custom properties.
Send contract notifications	Midnight daily	`NotificationScheduler.exe`	
Update FlexNet Manager Suite software usage history	4am daily	`BatchProcessTask.exe run FNMPSoftwareUsageHistoryUpdate`	Take a snapshot of all current licenses for use in reporting (to improve reporting performance).

Introducing the Batch Scheduler

The batch scheduler for FlexNet Manager Suite runs on the batch server. It is responsible for receiving messages that request specific tasks, and placing these in an execution queue in such a way as to eliminate incompatibilities and (to some degree) optimize performance.

The batch scheduler executable is `BatchProcessScheduler.exe`, which runs as a service on the batch server. The interface to this service is a separate utility, available in two different forms that provide identical functionality other than interactivity:

- `BatchProcessTaskConsole.exe` provides a command-line interface which also displays the output of results

- `BatchProcessTask.exe` is used for execution directly from a scheduling tool (such as Microsoft Task Scheduler), and as such accepts the same command-line inputs as `BatchProcessTaskConsole.exe`, but suppresses output to the console.

The majority of this documentation concerns the operation of either of the `BatchProcessTask[Console].exe` utilities, since these provide the command line interface for the batch scheduler.

How Batch Scheduling and Processing Works
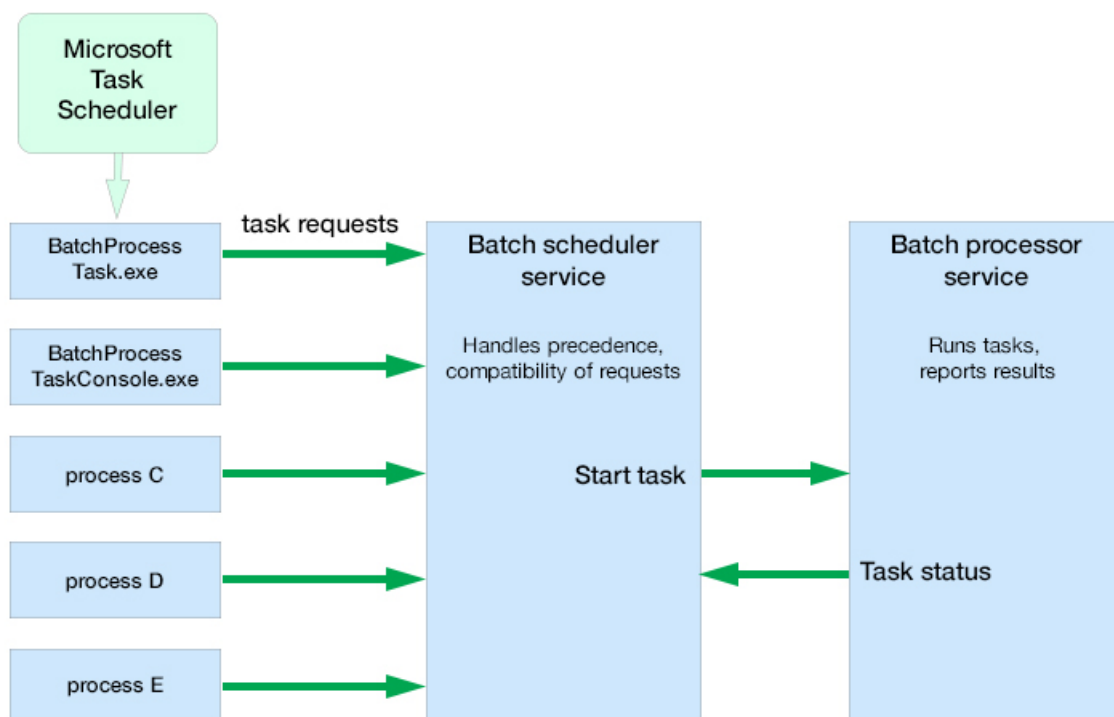
Batch scheduling and processing, summarized in the diagram above, runs as follows:

1. The batch scheduler service receives a message requesting a particular task. Messages may come from (among other sources):

   - `BatchProcessTask.exe` or `BatchProcessTaskConsole.exe` (including command-line input)

   - The web interface for FlexNet Manager Suite (for example, when an operator has uploaded spreadsheet or other data)

- The `ManageSoftRL` endpoint (the web service that receives uploads FlexNet inventory from inventory beacons)

- The `inventory-beacons` endpoint (used by inventory beacons to collect policy and updated rules, as well as to upload third-party inventory)

- The Activation Wizard (where you enter details of your license to use FlexNet Manager Suite).

2. The message is validated with the following tests:

- The message type is one of the supported batch processor tasks. If not, the failure is logged. (All supported tasks are listed in *Batch Scheduler Command Line* on page 156.)

- The scoping is correctly supplied, as listed in *Batch Scheduler Command Line* on page 156 (for example, if a group identifier is required, one is supplied and a tenant ID is not supplied; or for tasks requiring a tenant ID, that the ID is correctly supplied, and no group is present). If the scoping is not correctly specified, the message is failed, and the failure is logged. For a standard (single tenant) on premises implementation, no tenant ID is required for any task.

- The message is not a duplicate of one already in the task queue. A message is a duplicate if it has the same task type, group name, tenant UID, and parameter set (both names and values). Duplicates are discarded, and logged with the fact that they are duplicates.

3. Validated messages are saved to the database. This allows for recovery after a disastrous failure, such as the server suddenly going down.

4. Saved, valid messages are added to the in-memory pending messages queue.

5. The scheduler service regularly assesses the set of currently pending messages (from the oldest to the most recent) against any other tasks that are currently executing. The constraints include:

- Limits where no more than one of a particular task may be running at a time

- Mutual exclusions, where task B may not start while task A is running.

For the list of constraints, see *Batch Processing Constraints* on page 168.

6. When the scheduler decides a task is safe to execute, it sends a message to the batch processor.

7. The batch processor collects the task from the head of the queue, and executes it, returning a message to update the task status as required.

The possible status values that may be updated by either the batch scheduler or the batch processor (and are visible in the log files) are:

Status	Updated by	Description
Duplicate	Batch scheduler	New incoming task is a duplicate of an existing task that has a `Submitted` status. Duplicate entries have the same • Task type • Group name • Tenant UID • Parameter set.

Status	Updated by	Description
`Submitted`	Batch scheduler	Task submitted and is waiting (possibly on other tasks) for dispatch to the processor service.
`Queued`	Batch scheduler	Task dispatched to the tasks queue by the scheduler service, but not yet picked up by the processor service.
`Processing`	Batch processor	Task accepted by the batch processor service, and processing is in progress.
`Success`	Batch processor	Processor service reported the successful completion of the task (this may be reporting the return code from a child process).
`Error`	Batch processor	Error while processing task.

8. While processing, the batch processor continues to send 'heartbeat' messages to the batch scheduler (the heartbeat continues whether the batch processor is busy or idle).

Tip • *If the heartbeat messages fail for a specified period (default: 5 hours), all tasks recorded as* `Processing` *are marked as failed, and given the status* `Error`. *This may happen, for example, if the batch processor service was killed and failed to reappear.*

9. When the processing of the task completes, the status is updated to either `Success` or `Error`. Results are logged in `%ProgramData%\Flexera Software\Compliance\Logging\BatchProcessScheduler \BatchProcessScheduler.log` on the batch server (or the server hosting this functionality), with the log entry including text like the following:

```
received status update for task 'Success' from processor serverName
```

The same log message includes a task identifier in the following format:

```
Task[159add91-a5e4-4755-9ff4-d514baae2e11]:
```

To identify the type of task, you can:

• Search for the first occurrence of this ID in the batch scheduler log, such as:

```
Message DataWarehouseUpdateRights[159add91-a5e4-4755-9ff4-d514baae2e11]
sent for processing
```

• Search for the first occurrence of this ID in the `BatchProcessor.log` (in the same folder), which also lists the full command that the batch processor executed:

```
DataWarehouseUpdateRights[159add91-a5e4-4755-9ff4-d514baae2e11]
Starting process with command line '"C:\Program Files (x86)\Flexera
Software\FlexNet Manager Platform\DotNet\bin\ComplianceReader.exe"
-us false -importtype Exporters -e BusinessIntelligenceRights'
```

A by-product of this process is that it is not possible to schedule a particular task at a *precise* time. Scheduled times really mean "not earlier than" and "as close as possible to", and the actual execution start time depends on there being no constraints from other tasks to delay execution.

Using the batch scheduler to organize tasks ensures that:

* Processes for importing the product libraries (such as the Application Recognition Library), which exert locks on various database tables, do not cause failures in inventory and other kinds of imports which may access the same tables

* Imports automatically avoid each other, instead of failing if they are accidentally run at the same time

* Business and third-party inventory imports uploaded by an inventory beacon are automatically processed as they arrive at the central application server. In multi-tenant environments, the batch scheduler can maximize the parallelism between imports for different tenants.

* Tasks originating from different sources (listed at the start of this topic) can be executed on demand when possible, without causing random failures through accidentally requesting incompatible processes at overlapping times.

Rapid Processing of Third-Party Inventory

The batch scheduler provides some useful specializations for handling third-party inventory imports from disconnected inventory beacons (those that are not co-located on your central application server, or batch server in a multi-server implementation). The purpose of these specializations is to make third-party inventory results available in the web interface as promptly as possible.

The overall process is as follows:

1. An inventory beacon collects inventory from a third-party tool (such as Microsoft SCCM or IBM's ILMT).
2. The inventory data package is saved with an XML manifest that identifies the connection used for the inventory import (both by the name you defined for the connection, and by a GUID tying the connection to the particular inventory beacon).
3. The inventory package is immediately uploaded to a web service on the central application server (or the batch server in a large implementation with separate servers).
4. The web service queues an `InventoryImportReaders` message to the batch scheduler. (Details of task messages are included in *Batch Scheduler Command Line* on page 156.)
5. The reader task is scheduled as soon as constraints and current tasks allow, resulting in the third-party inventory reaching the internal staging tables.
6. After the success of that task, by default the batch scheduler queues an `InventoryImportWriters` message so that the data is also transferred into the operational tables in the compliance database.

   This chaining of the writer task is controlled by another specialized option on the command line of the batch scheduler:

   ```
   -p "ChainWritersMessage=true"
   ```

   The value of this Boolean is derived from a per-tenant setting in the database, called `PackageUploadTriggersWriters` (stored in the `ComplianceTenantSetting` table). Its effect is to cause the batch scheduler to queue tasks for the compliance writers immediately on the success of the import

readers task. These not only transfer the inventory to the operational tables in the compliance database, but also complete a new calculation of license consumption (reconciliation) based on this latest data. Therefore the inventory, and its impact on licensing, are available as quickly as possible in the web interface of FlexNet Manager Suite.

Tip • The same setting may be used in a single-tenant on premises implementation.

While you can chain the writers for all inventory sources, it doesn't have the same impact when used in the case of inventory collected by FlexNet inventory agent. This is because the individual `.ndi` files have over time been uploaded into the internal inventory database, and there is no equivalent "instant trigger" to cause this data source to be 'immediately' written into the compliance database (really, 'immediately' has no useful meaning when `.ndi` files continue to be uploaded to the inventory database at relatively random times).

In contrast, for third-party inventory, the specialized process described above means that third-party inventory is available in the web interface of FlexNet Manager Suite as promptly as possible after it is collected by an inventory beacon.

Configuration for Batch Processing

Single or multiple servers (and network share)

The batch scheduler service and the batch processor service are implemented on a single server, known as the batch server. (When the implementation is quite small, the batch server may also be combined with the inventory server, and potentially also the web application server; but for the moment, our focus is on batch processing.)

Communications between the batch server and the batch scheduler are local to the batch server, and the staging folder for data incoming from inventory beacons is on the same server. The default location is `%ProgramData%\Flexera Software\Beacon\IntermediateData`. This default is formed by appending `IntermediateData` to the value of the base directory saved in `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\ManageSoft Corp\ManageSoft\Beacon\CurrentVersion\BaseDirectory`. This base location is also used by other processes, and should be changed only with care.

Tip • A second folder, a network share, is used for handing off files uploaded through the web interface (such as inventory spreadsheet imports) for processing by the batch server. For this share, the default path is `%ProgramData%\FlexNet Manager Platform\DataImport`, and the path is saved in the registry at `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\ManageSoft Corp\ManageSoft\Compliance\CurrentVersion\DataImportDirectory`. There is also a parallel folder for data export. For implementations that separate the web application server from the batch server, these shares must also be configured and accessible from both servers. For more information, see Configure Network Shares for Multi-Server.

Installation and upgrade

The messaging that drives the batch scheduling and batch processing is implemented using Microsoft Message Queuing (MSMQ). In a multi-server implementation, MSMQ must be enabled on all servers. The MSMQ priority queues exist only on the batch server. For that reason, where other servers are separate, they must know the

fully-qualified domain name of the batch server so that they can access the queues. (Where there is only a single, combined application server, `localhost` may be used in place of the fully-qualified domain name of the server.)

These details of configuration are normally set up by PowerShell scripts during installation or upgrade of FlexNet Manager Suite. If at any stage there are new features installed, the PowerShell scripts should be re-run to update the configuration.

💡

Tip • *The name of the batch server is saved in the* `ComplianceSetting` *table of the compliance database, as* `BatchSchedulerHostName`.

Authentication and authorization

The batch scheduler and processor services must be executed using a valid account in Active Directory. During installation, through the PowerShell scripts, this same account is made a member of the Operator role (given full operator access to the system data). In a multi-server implementation, it is normal for the same service account runs on all the central servers, simplifying your administration of the message queues in MSMQ.

Authentication between the web application server and the batch server, or between any inventory beacon and the batch server, is handled using Windows authentication managed by MSMQ.

On the batch server (or, in a single server implementation, the application server), the account that runs the batch processor service was set up during implementation (the suggested value was `svc-flexnet`). This account must have inheritable permission to read the `%ProgramData%\FlexNet Manager Platform \DataImport` folder and all its sub-folders. (This is the default path: you can confirm the setting for your server by checking the registry at `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\ManageSoft Corp \ManageSoft\Compliance\CurrentVersion\DataImportDirectory`.) Typically these permissions are controlled through Active Directory group memberships, and you can check the permissions like this:

1. In Windows Explorer, right-click on the `DataImport` folder, and select **Properties** from the context menu.

2. In the **DataImport Properties** dialog, select the **Security** tab, and then click **Advanced**.

3. In the **Advanced Security Settings for DataImport** dialog, select the **Effective Permissions** tab.

4. Next to the **Group or user name** field, click **Select...**.

5. In the **Select User, Computer, Service Account, or Group** dialog, click **Object Types...**, ensure that **Service Account** is selected, and click **OK**.

6. In the **Enter the object name to select** field, enter the name of the account running the batch processor service (the name proposed during implementation was `svc-flexnet`). You can click **Check Names** to ensure that the account name is valid and recognized. Then click **OK** to return to the previous dialog, and display the permissions for this service account on the folder. (Since the service account is the same user context as runs the compliance readers or the Business Importer, the necessary rights are more extensive that required just for messaging.) As a minimum, the following permissions are required:

   • `List folder / read data`

   • `Create files / write data`

   • `Create folders / append data`

- `Delete subfolders and files`

- `Delete.`

7. These rights must be inheritable by any child objects (such as subfolders) that are created. In general, check the **Permissions** tab of the **Advanced Security Settings for** *Folder name* dialog. If it shows a checked (ticked) box for **Include inheritable permissions from this object's parent**, it typically also means that the inheritance property is also inherited. Otherwise, inheritance must be configured within Active Directory.

Tip • Microsoft IIS is configured by default not to use account impersonation. If IIS is reconfigured to use impersonation, MSMQ and folder permissions will need to be adjusted such that all system users have appropriate rights to the folders and the MSMQ incoming queue.

Limiting Parallel Execution Across Tenants

This process applies only to multi-tenant implementations (such as for managed service providers).

In a multi-tenant environment, it is possible for many tasks to be requested at once, potentially creating load issues on the batch processor. Some limits are in place naturally: for example, execution of the `ComplianceReader` is generally restricted to one-per-tenant. However, in a system with many tenants, the worst case is to have each tenant running one instance, and all running at once; and there are other tasks like SAP imports that also create load on the system.

There is a database feature that allows limits to be applied to sets of batch processing task types. For example, by default a limit named `ComplianceReader` is given a value of `5`. This limit is associated by ID with the following batch processing tasks:

- `ActiveDirectoryImport`

- `DataWarehouseUpdate`

- `EntitlementRecommendations`

- `InventoryImport`

- `InventoryImportNoStats`

- `InventoryImportReaders`

- `InventoryImportWriters`

- `LicenseReconcile.`

This means that across all tenants, there is a maximum of five of any of these tasks executing at once.

Tip • A task type may only be associated with at most one limit.

To create a limit:

1. In Microsoft SQL Server Management Studio, in the compliance database, insert a new row into the into `BatchProcessTypeLimit` table, with a unique `Name` and the upper limit set as the `Max Tasks` value, noting the `BatchProcessTypeLimitID` created for this record.

   For details, see *FNMSSchemaReferencePDF*, available through the title page of online help.

2. Update the `BatchProcessType` table, selecting the row with the appropriate `TypeName` for the task in question, and inserting the `BatchProcessTypeLimitID` value into the column of the same name for that row.

3. Repeat the last step for any other task types that should participate in the same limit.

Important • *The `IBMPVULicenseUpdate` task must not participate in a limit that involves other batch process tasks. Furthermore, if `IBMPVULicenseUpdate` is limited (on its own), the limit must be set to a high value. Tight restrictions on `IBMPVULicenseUpdate` may mean that FlexNet Manager Suite cannot meet the frequency of PVU checks required for IBM audit data and compliance.*

Batch Scheduler Command Line

Command-line reference for the server-side batch scheduler.

The command line syntax and results are identical for the two forms of the command-line utility for the batch scheduler service (`BatchProcessTaskConsole.exe` for interactive use, and `BatchProcessTask.exe` for programmatic use). These utilities exist solely for interpreting input and sending messages to the batch scheduler service.

Note • *The command-line syntax documented below works in the Windows command window. These commands are not suitable for use in the PowerShell interface.*

When you input a command (either through the console, or programmatically through a scheduled task)::

- `BatchProcessTask[Console].exe` validates the command line (this requires that the task name is valid, and parameters are specified appropriately for the scope of the task). Valid command lines are sent as a message to the batch scheduler service.

- The batch scheduler service checks against the list of pending tasks for duplicate messages (duplicates have the same task, group name, tenant UID, and parameter set). When a duplicate is detected, the original queued task is preserved, and the new command is not queued, but is saved in the database with the state `Duplicate`.

- The batch scheduler service persists the command to the database (protecting against unexpected shut-down, so that the system resumes cleanly after a restart).

- The batch scheduler service adds the command to the pending queue. It reviews this queue (from oldest to latest) regularly, and passes any task that is free of restrictions in a message to the batch processor service.

For more details, see *How Batch Scheduling and Processing Works* on page 149.

Synopsis

`BatchProcessTask[Console].exe run` *taskName* [*options*] [`--`*taskArguments*]

Tip • The default path to the executable is `C:\Program Files (x86)\Flexera Software\FlexNet Manager Platform\DotNet\bin`*.*

The leading set of options have limited applicability, and are uncommon for on premises implementations:

`-g` *groupName*   (applicable only to very large, multi-compliance-database implementations)
`-p`
`-t` *tenantUID* (applicable only to multi-tenant implementations, for managed service providers)

-g	*groupName*	Not relevant to, and must not be used with, on premises installations. For multi-database implementations, this means to run the task for all tenants in the specified group.
		Note • With this option, a tenant UID must not be specified in the command line.
		Groups apply only to very large implementations where the compliance data is split over multiple databases. The group identifies which database contains data records for the set of tenants. For on premises implementations, this option must be omitted.
-p		Pass properties to the batch processor that customize its behavior for certain tasks. These properties are not passed through as arguments to any executable that the batch processor may call in a sub-process. (For that purpose, use the `-` *taskArguments* format.) Instead, these properties modify the behavior of the batch processor itself. Properties must be specified in *name=value* pairs, and multiple pairs may be separated by commas.
-t	`TenantUniqueID`	In a multi-tenant environment, runs the task only for the tenant specified by the unique identifier. (This alphanumeric code is viewable in the Flexera Software license details, and in the `Tenant` table in the database.)
		Note • If a task is scoped to a tenant, and the `-t` option is omitted on the command line, `BatchProcessTaskConsole.exe` queues a separate message for every tenant in the database. These messages are then prioritized and processed as usual by the batch scheduler service.
		For single-tenant on premises implementations, this option should be omitted, and has no effect.

Tasks and Task Arguments

The following table lists the available task names, together with the *scope* of each task. The scope (System, Group, or Tenant):

- Specifies who is affected by an instance of the task (for example, `ActiveDirectoryImport`, which has a scope of Tenant, imports the AD information for a single tenant). System tasks are global, and must not have either a group name or a tenant ID specified. For an on premises implementation, Group scope is equivalent to System scope.

- Maps to the options described above, so that, for example, in a multi-tenant implementation for a managed service provider, `ActiveDirectoryImport` requires that the `-t` *TenantUID* option is specified to the batch scheduler service (if omitted at the command line, `BatchProcessTaskConsole.exe` queues a separate message for every tenant in the database.

- Describes the level of isolation for each task. For example, the `ActiveDirectoryImport` task cannot be run in parallel (that is, only one instance may run at a time) within its scope of a single tenant. However, in a multi-tenant environment, separate tenants may be running this task at the same time, as long as for each tenant exactly one instance of the task is running. (For more details about restrictions and constraints about what cannot be run in parallel, see *Batch Processing Constraints* on page 168)

The task arguments are not required for standard operation. The batch processor service adds a standard set of arguments for each task (in the table, shown with the "Underlying executable" entry). If you are customizing behavior, you may add additional task arguments, which are appended to the standard set (you cannot override the standard arguments). If you are adding task arguments, each set must be preceded by *two* dashes ('minus minus'), which indicates that the argument(s) are to be passed through directly to the executable invoked when processing the task. The set of arguments must then be enclosed in double quotation marks if they contain any white space.

Task	Scope	Notes
`ActiveDirectoryImport`	Tenant	Import sites, subnets, computers, groups, and users from Active Directory. There should be no need to trigger this task manually, nor to schedule it separately, as the appropriate message is queued by `mgsimport.exe` after importing an uploaded file to the inventory database. 💡 *Tip • When Active Directory data is uploaded from an inventory beacon, a Microsoft scheduled task triggers its import into the inventory database (this task by default runs every 10 minutes). At the successful completion of this import, a message is queued for the batch scheduler to schedule* `ActiveDirectoryImport` *for the connection to the inventory database. The bottom line is that Active Directory data is available in the web interface of*

Task	Scope	Notes
		FlexNet Manager Suite as quickly as possible after it is scheduled for collection on any inventory beacon. Underlying executable: ``` ComplianceReader.exe -s connectionName -e ActiveDirectory ```
ActivityLogHistoryDelete	Tenant	Clean up any records in the system activity log that are older than (by default) 90 days. Underlying executable: SQL stored procedure.
ARLCleanup	System	This task cleans up downloaded library files (both .cab files and extracted files) after the contents have been imported into FlexNet Manager Suite. 💡 *Tip • It is not necessary to schedule or execute this task. In normal operations, the batch scheduler automatically schedules a cleanup after all required imports are completed.* Underlying executable: ``` MgsImportRecognition.exe -c ```
ARLDownload	System	Downloads the latest version of the Application Recognition Library, saving the contents on the batch server for later import (using the separate ARLImport task). 💡 *Tip • If you are managing your own scheduling, you do not need to separately schedule the following ARLImport and ARLCleanup tasks. The batch scheduler automatically queues these tasks after a successful download.* 📋 *Note • In a multi-tenant environment for managed service providers, the Application Recognition Library is downloaded once to a shared location, and is then available for import to the system.*

Task	Scope	Notes
		Default schedule: Daily at 1am local time. Underlying executable: `MgsImportRecognition.exe -dl` 💡 *Tip • It is important that you use the batch scheduler to run the library downloads and imports. Other imports run at unpredictable times: inventory imports, business imports, and data exchange with FlexNet Manager for Engineering Applications and FlexNet Manager for SAP Applications are all run on demand as data is uploaded by inventory beacons. As well, some imports may be requested by operators using the web interface. If you use the batch scheduler to download the libraries, the batch scheduler will gracefully pause these competing tasks, or wait for their completion, before pushing through the content update, and then resume processing other tasks afterward. In contrast, if you do not use the batch scheduler and run the process directly from the `MgsImportRecognition.exe` command line, the database locking required for the library updates may mean that either the library update itself, or a competing task, will fail.*
`ARLImport`	Group	Import the last downloaded Application Recognition Library, currently saved on the local disk. Also download and import the SKU library, and the individual PURL libraries as authorized by your license terms. 💡 *Tip • For normal operations, do not schedule (or execute) this task. In normal operations, the batch scheduler automatically schedules this task after the download of the ARL is completed.* 📋 *Note • In a multi-tenant environment for managed service providers, the SKU and PURL libraries are downloaded once, and the logical-AND of all tenant entitlements is loaded into the database. Access by*

Task	Scope	Notes
		individual tenants is then managed by their license terms.
		Underlying executable:
		```MgsImportRecognition.exe -g groupName -ia -ne```
`BaselineImportProcessing`	Tenant	Used for processing imports of Microsoft Licensing Statements (MLS). Not available for external use.
		Underlying executable: SQL stored procedure.
`BusinessAdapterConfig`	Tenant	Updates the data model (`FNMPDataModel.ini`), the DDI files, and the CSV templates used by the Business Importer (and therefore the Business Adapter Studio). The updated content is downloaded to the inventory beacons the next time they request an update (by default, every 15 minutes, and configurable through **Discovery & Inventory** >**Settings**, in the **Beacon settings** section). This update is particularly valuable for keeping the Business Importer up to date with changes to custom properties. For more information, including details of the standard data model and CSV templates, see *Using the FlexNet Business Importer*, a PDF file available through the title page of the online help. In a multi-tenant implementation, this may be run for a single tenant (with the tenant UID specified), or may be run without the `-t` option to update all tenants.
		Default schedule: Daily at 4am local time. Also triggered when a new tenant license is imported (in a multi-tenant implementation).
		Underlying executable: SQL stored procedure.
`BusinessAdapterImport`	Tenant	Imports any packages uploaded by the Business Importer to the central application server, saving the data in the compliance database ready for the next calculation of license consumption (which must be triggered separately).
		Note • Although the scope of this task is per tenant (the data for only one tenant is imported at a time), the task can only run one-at-a-time within a group, as logging is shared for all tenants in a group.

Task	Scope	Notes
		Underlying executable: `mgsbi.exe`
`DataWarehouseUpdate`	Tenant	Update the data warehouse with the latest information (to allow for trend reporting and the like). Default schedule: Weekly on Sunday at 6am. Underlying executable: ```ComplianceReader.exe -e BusinessIntelligenceRights -e BusinessIntelligenceData```
`DataWarehouseUpdateRights`	Tenant	Copy the tenant list, and update the access rights to the data warehouse. Default schedule: triggered when a new tenant license is imported. Underlying executable: ```ComplianceReader.exe -us false -importtype Exporters -e BusinessIntelligenceRights```
`EnterpriseGroupImport`	Tenant	Used for imports through the web interface. Not available for external use. Underlying executable: SQL stored procedure.
`EntitlementRecommendations`	Tenant	A task triggered through the web interface. Not available for external use. Underlying executable: ```ComplianceReader.exe -us false -e EntitlementAutomation```
`FNMEAEnterpriseGroupExport`	Tenant	Transferring current enterprise group structure for use in FlexNet Manager for Engineering Applications. Not available for external use. Underlying executable: SQL stored procedure.
`FNMPDataMaintenance`	Tenant	Internal database maintenance. Not available for external use. Underlying executable: SQL stored procedure.
`FNMPSoftwareUsageHistoryUpdate`	Tenant	Internally, takes a snapshot of all licenses to improve reporting performance. Not available for external use. Underlying executable: SQL stored procedure.

Task	Scope	Notes
IBMPassportAdvantage	Tenant	Used for import of IBM Passport Advantage reports through the web interface. Not available for external use. Underlying executable: SQL stored procedure.

IBMPVULicenseUpdate	Tenant	Used for high-frequency tracking of PVU licenses. Triggered internally, and not available for external use.
		
		Task arguments:
		
		```"-e Computer -e VirtualMachine -qualifiedsource \"ConnectionNameAndGUID\" -us false -e LicenseReconciliation -concurrent IBM -overrideparams PublisherFilter=IBM -processremotepackages false"```
		
		Underlying executable:
		
		```ComplianceReader.exe -e Computer -e VirtualMachine -qualifiedsource \"ConnectionNameAndGUID\" -us false -e LicenseReconciliation -concurrent IBM -overrideparams PublisherFilter=IBM -processremotepackages false```

IMDataMaintenance	Group	Executes a database stored procedure for data maintenance within the FlexNet inventory database across all the tenants in the group. Underlying executable: SQL stored procedure.
IMImport	Tenant	Runs the inventory import for FlexNet inventory (collected by the FlexNet inventory agent, either locally installed on the inventory device or taking zero-touch inventory from an inventory beacon). The data is taken from the inventory database (regarded now as a data source), and copied into the staging tables within the compliance database by the inventory reader. It then queues a `InventoryImportWriters` task. By default, this import occurs daily at midnight. This task is intended for tenants in the cloud-based implementation. For these tenants, any third-party inventory is collected by disconnected inventory beacons (those not installed on the central application servers), and immediately uploaded, and imported on demand. For such tenants, only the FlexNet inventory remains to be imported prior to queueing the writers. Underlying executable: `ComplianceReader.exe`

Task	Scope	Notes

InventoryImport	Tenant	This task is a single invocation for a complex of inventory import and license consumption calculations, in a single convenience command for on premises implementations only. With the first form of task arguments shown, inventory from all available connections is imported and processed. Export to the data warehouse is specifically excluded. You can also restrict this command to a single inventory source with the `-s` task argument, followed by the connection name. For example, adding
		
		```---s """FlexNet Manager Suite"""```
		
		will restrict processing to the default connection for inventory gathered by the FlexNet inventory agent and saved in the internal inventory database.
		
		This task is intended for use on premises, where it is possible that there are third-party inventory connections on an inventory beacon co-installed on the batch server. For these connections, there is no intermediate staging file, and no import on demand (as there is for connections exercised on disconnected inventory beacons). For this reason, all locally-available connections are imported before also importing FlexNet inventory, and queueing the writers.
		
		Default schedule: Daily at 2am local time.
		
		Underlying executable:
		
		```ComplianceReader.exe -us false -d BusinessIntelligenceRights -d BusinessIntelligenceData -t tenantUID```

InventoryImportNoStats	Tenant	This task is queued when an operator (in the `Administrator` role) selects the **Update inventory** check box and clicks the **Reconcile** button in the web interface. The name arises because this process, for speed, skips updating database statistics. Underlying executable: ```ComplianceReader.exe -us false -d BusinessIntelligenceRights -d BusinessIntelligenceData -it ReadersAndWriters -us false```

Task	Scope	Notes

`InventoryImportReaders`	Tenant	By default, runs the import from all data connections (and all pending data packages) specified for the tenant into the staging tables within the compliance database. For inventory collected by FlexNet inventory agent, the data that has been uploaded into the internal inventory database is imported (that is, the inventory database is treated as another connection).
		
		The batch scheduler receives another specialized property with this task message:
		
		`-p "ChainWritersMessage=true"`
		
		This means that the writer tasks are queued after the success of this task, so that third-party inventory in particular is available as soon as possible (for details, see *Rapid Processing of Third-Party Inventory* on page 152).
		
		Default schedule: None (instead, for scheduled imports, see `InventoryImport`).
		
		Underlying executable:
		
		`ComplianceReader.exe -us false`
		`-d BusinessIntelligenceRights`
		`-d BusinessIntelligenceData`
		`-importtype Readers`

`InventoryImportSpreadsheet`	Tenant	Queued when an operator performs a one-off inventory spreadsheet upload through the web interface. Not available for external use.

`InventoryImportWriters`	Tenant	Takes the data currently available in the staging tables within the compliance database, de-duplicates records as necessary, writes the results into the main data tables of the compliance database, and calculates the resulting license consumption.
		
		Default schedule: None (instead, see `InventoryImport`).
		
		Underlying executable:
		
		`ComplianceReader.exe -us false`
		`-d BusinessIntelligenceRights`
		`-d BusinessIntelligenceData`
		`-importtype Writers`

`LicenseReconcile`	Tenant	Runs only the license consumption calculations of the compliance writers (that is, does not include writing the

Task	Scope	Notes
		last data from the staging tables into the compliance database). Underlying executable: ```ComplianceReader.exe -us false -e LicenseReconciliation```
`POLineImport`	Tenant	Queued when purchases are imported through the web interface. Not available for external use. Underlying executable: SQL stored procedure.
`SAPInventoryImport`	Tenant	Import inventory from FlexNet Manager for SAP Applications. Default schedule: Daily at 10pm batch server time. Underlying executable: ```SAPReader.exe -b```
`SAPPackageLicenseImport`	Tenant	Import the licenses calculated by SAP, writing results into the compliance database. Default schedule: Weekly on Sundays at 6am. Underlying executable: ```ImportPURL.exe -l SAPPackages```
`SAPTransactionProfileImport`	Tenant	Internal use. Not available for external use. Underlying executable: SQL stored procedure.
`SAPUserAndActivityImport`	Tenant	Internal use. Not available for external use. Underlying executable: SQL stored procedure.
`SAPUserConsumptionImport`	Tenant	Internal use. Not available for external use. Underlying executable: SQL stored procedure.
`SAPUserImport`	Tenant	Internal use. Not available for external use. Underlying executable: SQL stored procedure.
`SAPUserRecommendationsExport`	Tenant	Internal use. Not available for external use. Underlying executable: SQL stored procedure.
`SAPUserRoleImport`	Tenant	Internal use. Not available for external use. Underlying executable: SQL stored procedure.

Task	Scope	Notes
ServiceNowExport	Tenant	Export data to ServiceNow, based on the integration set up between ServiceNow and FlexNet Manager Suite. May also be trigger by operator activity in the web interface. Default schedule: Weekly on Sundays at 3am. Underlying executable: `fnmp_servicenow_export.exe`
UserAssignmentImport	Tenant	Internal use. Not available for external use. Underlying executable: SQL stored procedure.

Additional Verbs for the Batch Scheduler

Other commands for the batch scheduler are useful for debugging and recovery work, in case of problems.

In normal operation, the batch scheduler almost exclusively makes use of the `run` verb, as described in *Batch Scheduler Command Line* on page 156. The additional verbs covered here are generally reserved for development, testing, and remediation.

`BatchProcessTaskConsole.exe help`

Prints a text message to the console showing the command-line options.

`BatchProcessTaskConsole.exe list-tasks`

Creates a list of tasks that are currently executing, together with requests for tasks to execute in future.

`BatchProcessTaskConsole.exe process-dispatch -p`

Sends a pause message to the batch scheduler service through the express (or process control) queue. Once the message is received and saved to the database, the batch scheduler service stops the queue processing and message dispatch to the batch processor. It writes a 'paused' record to the log file, and repeats this logging every minute until processing is resumed.

`BatchProcessTaskConsole.exe process-dispatch -r`

Sends a resume message to the batch scheduler service through the express (or process control) queue. Once the message is received and saved to the database, the batch scheduler service restarts the queue processing and message dispatch to the batch processor. It also stops writing paused records in the log file.

`BatchProcessTaskConsole.exe test` *taskName* [*options*] [--*taskArguments*]

Use with the same task names and arguments as listed in *Batch Scheduler Command Line* on page 156. This causes the batch scheduler to treat the request as usual; but when the message reaches the batch processor, it does not execute the task, but instead pauses for a brief period. Together with a subsequent examination of the log files, this test can help to ensure that the system is running correctly.

`BatchProcessTaskConsole.exe fail-task -m` *messageGUID*

This is an important but dangerous command, and should only be used when a task has been lost and the system has failed to heal itself, resulting in a backlog of tasks. This will manifest as the 'lost task' sitting in the processing state for a significant (multi-day) period, while the pending tasks are not progressing, even though actual processing on the task has finished. Use the `list-tasks` command to check the queue and running

tasks. It is bad practice to use the `fail-task` command to terminate a task that is still running normally, albeit for a long time. For further information, see *Troubleshooting Batch Processing* on page 171.

Batch Processing Constraints

The batch scheduler imposes the following constraints on tasks that are queued for the batch processor. (These collections or groupings of tasks are labelled 'bands' to avoid confusion with database groups in massive implementations.)

Constraints are of two types:

* Within each of the following bands, for the scope(s) described, only one task may run at a time.

* Between certain bands, as specified below, operations are mutually exclusive. For example, any running task in band A prevents any task in band D from starting.

Therefore, when `ActiveDirectoryImport` is running, it blocks all other tasks in band A (unique task within band) *and* all tasks in band D (mutually exclusive bands).

Band A: Inventory imports

For each tenant (on a multi-tenant system), or across the system (for an on premises implementation), only one of the following tasks can be run at a time:

* `ActiveDirectoryImport`

* `DataWarehouseUpdate`

* `DataWarehouseUpdateRights`

* `EntitlementRecommendations`

* `InventoryImport`

* `InventoryImportNoStats`

* `InventoryImportReaders`

* `InventoryImportSpreadsheet`

* `InventoryImportWriters`

* `LicenseReconcile`

In addition, `IBMPVULicenseUpdate` blocks band D (Content) in the same way as band A members do; but it is *not* mutually exclusive with the members of band A (Inventory imports).

Band B: Business imports

For each group (in a massive implementation with multiple compliance databases, or across the system (for an on premises implementation), only one of the following tasks can be run at a time:

* `BusinessAdapterImport`

* `EnterpriseGroupImport`

- `IBMPassportAdvantage`

- `POLineImport`

- `UserAssignmentImport`

Band C: SAP imports

For each tenant (on a multi-tenant system), or across the system (for an on premises implementation), only one of the tasks in band C or D can be run at a time:

- `SAPInventoryImport`

- `SAPPackageLicenseImport`

- `SAPUserAndActivityImport`

- `SAPUserRecommendationsExport`

Band D: SAP business imports

These tasks remain mutually exclusive with those in band C (SAP imports), but in addition are mutually exclusive with band B (Business imports):

- `SAPTransactionProfileImport`

- `SAPUserConsumptionImport`

- `SAPUserImport`

- `SAPUserRoleImport`

Band E: Content

Across the system (or across a group, for those massive implementations), only one of the following tasks can be run at a time:

- `ARLCleanup`

- `ARLDownload`

- `ARLImport`

Band F: General

For any individual tenant, only one instance of each of the following tasks can run at a time (no parallel instances of the same task, but these tasks are not mutually exclusive):

- `ActivityLogHistoryDelete`

- `BaselineImportProcessing`

- `BusinessAdapterConfig`

- `FNMEAEnterpriseGroupExport` (mutually exclusive with all tasks in band B - Business imports)

- `FNMPDataMaintenance`

- `FNMPSoftwareUsageHistoryUpdate`

- `IMDataMaintenance` (one running instance per group, where groups apply; and otherwise one per system for this task)

- `ServiceNowExport` (mutually exclusive with all tasks in either band A [Inventory imports] and band B [Business imports])

Mutually-exclusive bands

When two bands are mutually exclusive, it means that any task from either band blocks all tasks from the other band.

- Band A (Inventory imports) and band E (Content) are mutually exclusive

- Band D (SAP business imports) and band B (Business imports) are mutually exclusive

- Some members in band F (General), as marked, are mutually exclusive with tasks in bands A and/or B.

Separating Readers and Writers

The import and processing of inventory information is divided into two stages that are controlled by separate code entities within FlexNet Manager Suite:

- Compliance "readers" collect data that has been uploaded to the central application server, and write it into staging tables within the operations databases.

- Compliance "writers" take the data from the staging tables, normalizing it where required, and write the results into the operational tables in the compliance database. They perform recognition of newly-inventoried evidence against the ARL, and evaluate any pending automated purchase processing. Thereafter they recalculate license consumption (or 'reconciliation'), using the most recently updated data.

It is possible to run the compliance readers and writers in tandem, such that every 'read' is promptly followed by an immediate 'write' and reconciliation. However, the write and recalculate processes can be resource intensive, slowing performance for other tasks. Keeping in mind that inventory data may be incoming from multiple different sources (FlexNet inventory agent, third-party inventory tools, Active Directory, business imports, and so on), allowing a write and recalculate after every single data load may become problematic.

For this reason, the batch scheduler allows for triggering compliance readers and writers separately, completely independent of one another.

In its default operation, the batch scheduler handles this sequencing automatically.

However, if you are manually scheduling various processes, or running processes from the command line, you should be aware of the distinction between various tasks (described in *Batch Scheduler Command Line* on page 156). For example:

- This command imports inventory from the single default connection for inventory collected by FlexNet inventory agent, and immediately runs the writers (transferring data into the compliance database) and then running the license consumption calculations:

```
BatchProcessTaskConsole.exe run InventoryImport ---s """FlexNet Manager Suite"""
```

- This command loads the same inventory to the staging tables, but does *not* include the writing to the compliance database nor the license consumption calculations:

```
BatchProcessTaskConsole.exe run InventoryImportReaders ---s """FlexNet Manager Suite"""
```

You may then choose to run a number of other imports (readers) from other inventory connections, and only later run the writers and recalculation (see `InventoryImportWriters`).

If you are doing your own scheduling for these processes, you should allow the batch scheduler to minimize the number of writer and recalculation tasks that are run. This ensures that your automation integrates successfully with other imports that may be triggered by (for example) an upload from an inventory beacon or an import of a CSV file through the web interface. The way to give the batch scheduler that freedom is to add the `ChainWritersMessage` parameter to the readers command. For example, this command (executed by Microsoft Task Scheduler, and therefore no longer using the console version of the executable) runs all inventory imports, one after another as appropriate; and only when all are completed does it schedule the writers and recalculation:

```
BatchProcessTask.exe run InventoryImportReaders ---s """FlexNet Manager Suite""" -p
 "ChainWritersMessage=true"
```

Troubleshooting Batch Processing

The following notes may assist in troubleshooting the batch scheduler and batch processor on your batch server.

Examine log files

The batch scheduler and batch processor create log files under `%ProgramData%\Flexera Software \Compliance\Logging\BatchProcessScheduler`. Tasks go through the batch scheduler first, so you should first investigate `BatchProcessScheduler.log`. You can then use the ID given to each task to track processing results in `BtachProcessor.log`.

For a list of the status values that may be logged, and their meaning, see *How Batch Scheduling and Processing Works* on page 149.

Checking the state of batch processing

You can inspect the batch scheduler's view of what tasks are queued, executing, and recently completed by executing the following command:

```
BatchProcessTaskConsole.exe list-tasks
```

This command is equivalent to the following database query (which you could also use within Microsoft SQL Server Management Studio):

```
SELECT *FROM BatchProcessExecutionInfo bi
WHERE BatchProcessStatusID IN (1, 2, 3)
ORDER BY bi.StatusOrder, bi.DateOrder
```

For more information about using the `list-tasks` command in debugging, see *Killing or Failing Tasks* on page 174.

Checking the database record

You can use Microsoft SQL Server Management Studio to examine the contents of the
`BatchProcessExecution` table (for details of the table columns, see *FNMSSchemaReferencePDF*,
available through the title page of online help). There is also a convenient view on this table, called
`BatchProcessExecutionInfo`. This can be used to monitor the health of executing tasks, including those
requested by Windows Scheduled Tasks.

Additional troubleshooting

The following topics may also assist with troubleshooting.

Correct Identification and Connection

In a multi-server implementation, where the web application server is a distinct machine from the batch server, the
fully qualified domain name of the batch server must be available so that both servers can access the Microsoft
Message Queuing (MSMQ). The name is saved in the `ComplianceSetting` table of the compliance database,
as `BatchSchedulerHostName`.

If this record is incorrect, it is best practice to re-run the PowerShell installation scripts with the correct data.
However, you can also update the `ComplianceSetting` table directly.

In a multi-server implementation, on each server in turn, use the following command to determine the host name
and default DNS suffix of the current machine:

```
ipconfig /all
```

Ensure that you can successfully ping each server from all of the others. In particular for MSMQ, ensure that
from the web application server, you can ping the fully-qualified domain name stored for the batch server (and
matched in the output of the `ipconfig` command). Ping tests reachability as well as DNS resolution. If the ping is
not successful, remedy any DNS issues or barriers to connectivity (assuming that ping itself is not blocked).

If you make any change to the naming of the batch server, or change records of it available to other central
application servers, you must manually restart the batch processor and batch scheduler services. In addition, you
can either:

• Wait ten minutes for the ancillary IIS services to restart (this is the least disruptive in a production environment)

• Restart IIS manually (this restarting approach may be faster in a test environment).

Within about a minute of IIS restarting, the batch scheduler service has reconnected to the database, re-
established the message queue from the database record, and messaged the batch processor service to resume
processing.

Tip • *In a single-server environment, it is sufficient to specify* `localhost` *as the batch server name when
configuring other server functionality.*

Monitoring, Stopping, and Restarting the Services

It is important to monitor health of the batch scheduler and batch processor services. If they are not operational, FlexNet Manager Suite is not performing tasks that manage data integrity. The web interface continues to operate, but data that relies on background processes (such as license consumption calculations) is not updated until the processes are once again operational.

In the main, you should not need to intervene manually with these services. By design, both are set to automatically restart after failure, and this is an important part of the system's ability to heal itself. For example, if the database becomes inaccessible or the message queues are misconfigured, the batch scheduler responds by shutting down. Thereafter, the Windows Service restart setting allows the batch scheduler to try again each minute until processing can begin again.

Tip • This configuration is set during installation, and should not be changed without configuring a different method of restarting the service.

If you do need to restart the batch *scheduler*, previously requested tasks that are queued are preserved. These have been saved to the database as part of their processing (see *How Batch Scheduling and Processing Works* on page 149). On restart, the batch scheduler restores its internal state (the task queues) and continues as before.

If you restart the batch *processor*, running tasks are not left as orphans. On shut-down, the batch processor forcibly closes any processes that it is currently executing. These tasks are marked as failures in the execution history in the log file. Failed tasks are not resumed on restart. For this reason, it is best practice to use the following process for a shut-down and restart of the services:

1. Pause the batch scheduler to stop it sending new tasks to the batch processor:

    ```
    BatchProcessTaskConsole process-dispatch -p
    ```

2. Monitor the running tasks, repeating until no tasks are listed as executing:

    ```
    BatchProcessTaskConsole.exe list-tasks
    ```

    Tip • You can also review `BatchProcessExecution/BatchProcessExecutionInfo` *in the database.*

3. Stop the services and complete whatever adjustments are necessary.

4. Restart the services. The batch scheduler restarts in its paused mode.

5. Resume normal processing of the pending queue, and dispatch of tasks to the batch processor:

    ```
    BatchProcessTaskConsole process-dispatch -r
    ```

For more details of pausing and resuming batch scheduling, see *Additional Verbs for the Batch Scheduler* on page 167.

Killing or Failing Tasks

There is no direct way to kill tasks in the batch processing service. You may be able to interrupt and close executables called by the batch processor.

You can also 'fail' tasks; but misusing this feature in an attempt to interrupt and kill the task poses a considerable risk to system stability and data integrity. This is because failing the task only instructs the batch scheduler process to regard the task as dead, but does not stop the batch processor from running the task.

A task should only be failed when it is 'lost'. This term means that execution has finished in the batch processor, but the batch scheduler has not been notified of its completion. If the task imposes constraints (listed in *Batch Processing Constraints* on page 168), its coninued presence in the batch scheduler list may be blocking other tasks. In the very rare case of a suspected lost task, first validate that the batch processor is (still) configured properly to send messages to the batch scheduler; and then you can validate that a task is lost, and if necessary fail it, using these steps:

1. On your batch server, inspect the batch processor log files under `%ProgramData%\Flexera Software \Compliance\Logging\BatchProcessScheduler`.

2. If there are any error messages indicating that the batch processor cannot send messages to the batch scheduler, suspend this process, and first remedy the configuration of your batch scheduler and batch processor.

   Particularly for server identification issues, you may wish to re-run the PowerShell configuration scripts provided for installation and implementation, as these set up the correct relationship between the two services. You may alternatively need to repair permissions on the message queue; ensure MSMQ is running on all applicable servers; or take other action as required to remedy the particular problem. When these two services are again communicating properly, you may resume stepping through this process to clean up any task you suspect was lost during the period of messaging failure.

3. Execute:

   ```
   BatchProcessTaskConsole.exe list-tasks
   ```

   The output includes the message GUID, which can be used when searching the batch scheduler/processor logs. Depending on when the breakdown occurred in status messaging from the batch processor back to the batch scheduler, the suspect task may be shown as either `Queued` (if the batch processor hasn't yet picked up the task, or if the message that the batch processor started work did not get through) or `Processing` (if only the end-of-task success or failure message was lost). Any other value of the task status means that the task is not lost, and you should exit this process and review alternative explanations.

4. Re-examine the batch processor log file, now looking for evidence that the suspect task has indeed finished. Alternatively, in cases where the batch processor starts a child process, examine that process or its logs for evidence of completion.

   The underlying executables that are run in the child processes are listed in *Batch Scheduler Command Line* on page 156.

5. When the log files show the suspect task has definitely completed (successfully or otherwise, with mere completion being the requirement), and the `list-tasks` output still shows the same task as either `Queued` or `Processing`, copy the message ID from the output of the `list-tasks` command.

6. Substituting your copied message ID for the placeholder shown below, execute:

   ```
   BatchProcessTaskConsole.exe fail-task  -m messageGUID
   ```

The batch scheduler updates the status of the task to `Error` to mark the failure (you can verify this by re-running the `list-tasks` command shown earlier).

Any blocked tasks can now be queued by the batch scheduler, and then executed by the batch processor.

Symptom: ARL Starts While Blocked

Even though other (time-consuming) tasks are running that should prevent the execution of the `ARLImport` task (as listed in *Batch Processing Constraints* on page 168), a queued request for `ARLImport` may suddenly trigger the execution of the task.

This is because the `ARLImport` task has (by default) a 60-minute 'starvation limit' that prevents it being held up (or 'starved' of processing time) for too long. Before the time limit is exceeded, tasks with lower exclusivity requirements (tenant scope, compared with the system/group scope for `ARLImport`) are allowed to run, even if requested later than the `ARLImport` task, where the latter is being delayed by other tasks. When the starvation limit is exceeded, the batch scheduler tightens constraints, and no longer allows tasks with lower exclusivity to 'sneak in' to the queue. Furthermore, as soon as the `ARLImport` is starved, any arriving request that would normally block the `ARLImport` is itself blocked first. These changes prioritize the queued `ARLImport` task for execution by moving it quickly to the front of the execution queue (for operational details, see *How Batch Scheduling and Processing Works* on page 149). This preemptive move prevents the task sitting in the `Submitted` state for an excessive (even indefinite) time, blocked by other constraining tasks.

By default, only the `ARLImport` task has a starvation limit. The number of minutes is stored in the `StarvedAt` column of the `BatchProcessType` database table. The use of this table means that it is theoretically possible to set a starvation time limit for all types of batch processor tasks. However, setting such 'preemptive strikes' on the batch scheduler is not recommended, since it is only useful for tasks with high exclusivity (like system) being starved by tasks with lower exclusivity (like tenant). Best practice is to allow the batch scheduler to fulfill its prioritization tasks without prejudice.

6

Inventory Adapter Studio

Topics:

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.

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.

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 Software 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`).

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.
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 Software. - Steps may be added, deleted or have their execution order changed using the toolbar in the step explorer.
Step editing panel	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:
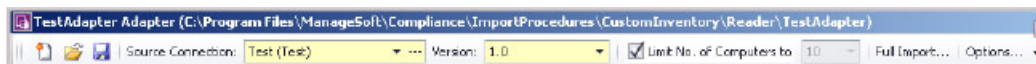
New button

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.
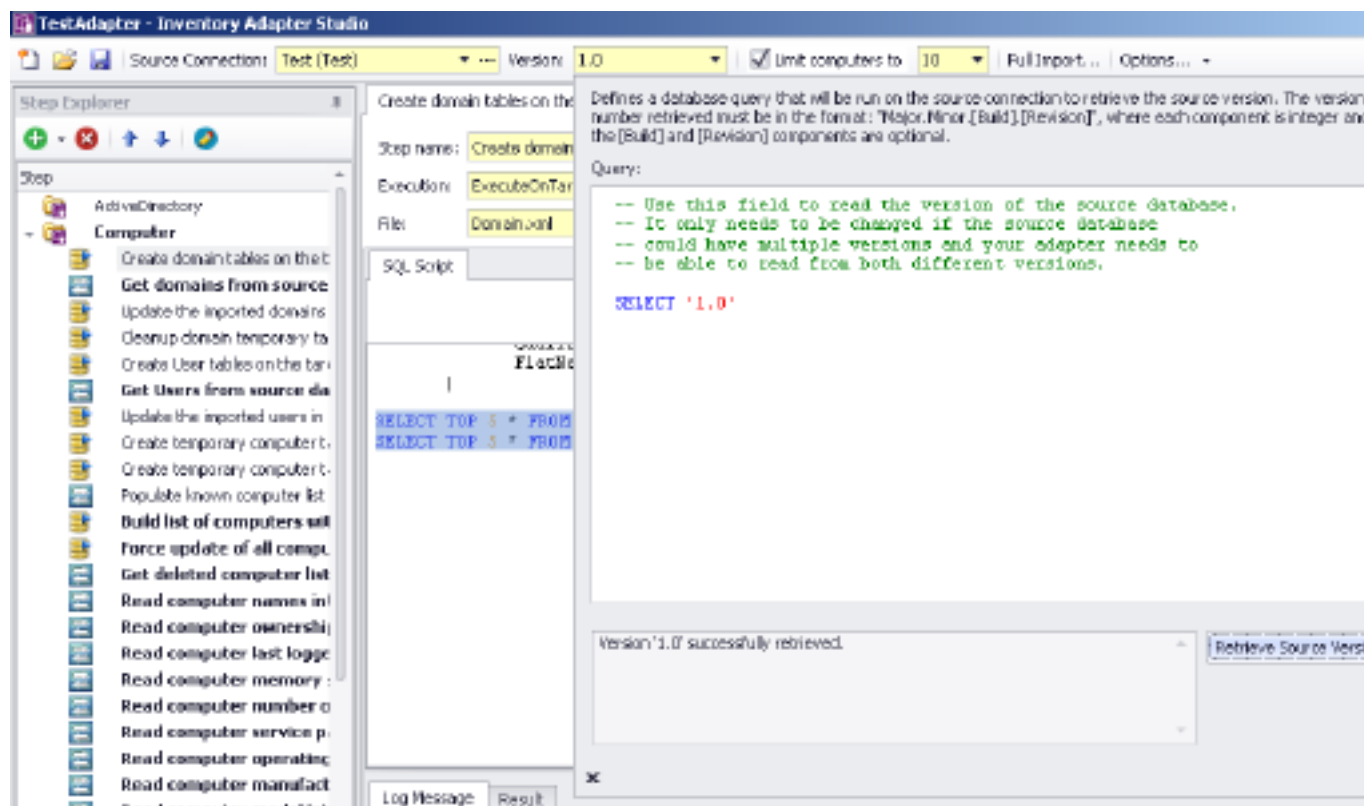
Use of this control is particularly important if your adapter supports importing inventory data from multiple versions of the same system. This usually occurs as enterprises upgrade systems over time.

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.
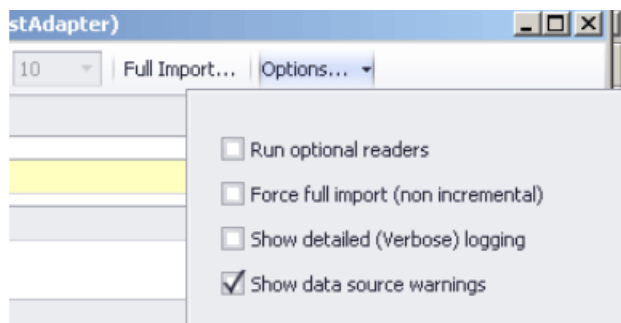
Options include:

- **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 greyed 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.
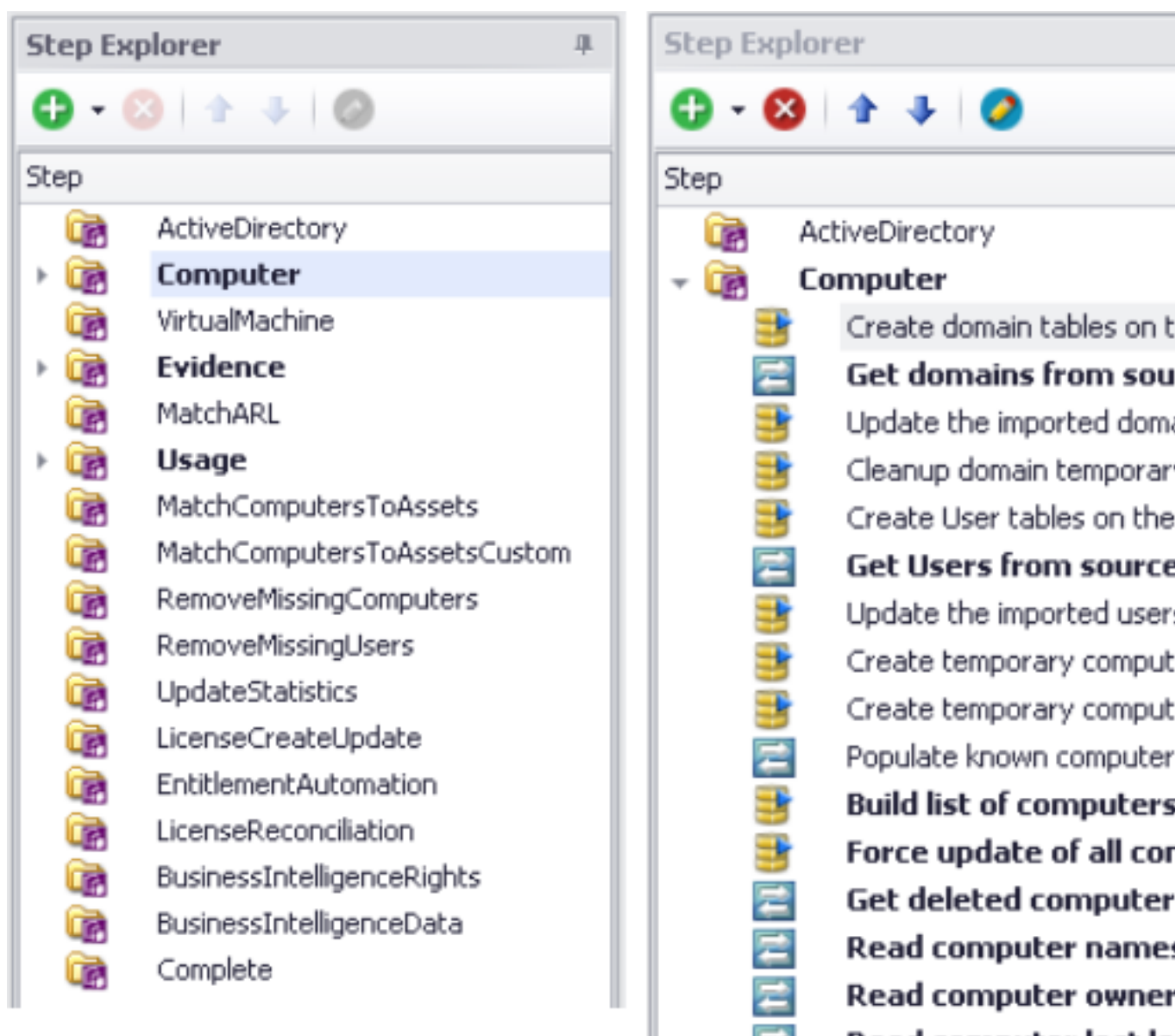
Figure 1: Collapsed (left) and expanded, with bold steps requiring customization. Custom SQL and data transfer steps visible.

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.
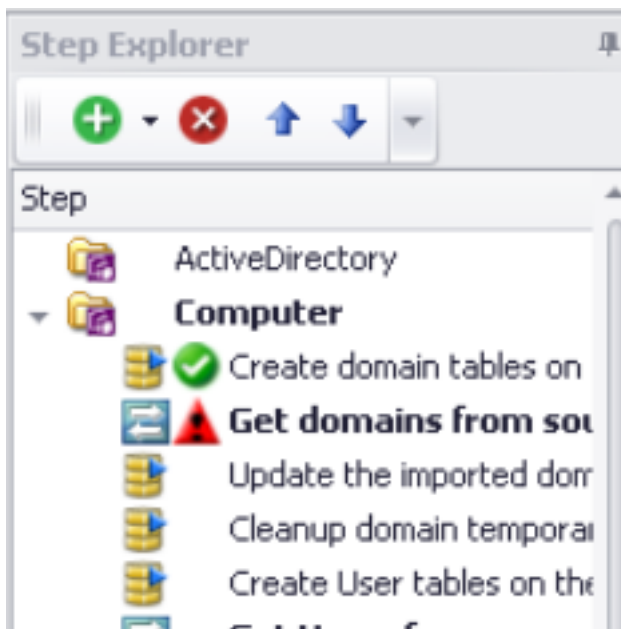
When the attribute values of a step will prevent it being executed for the version of the current connection, it will be greyed out with a strike-through in the step explorer.
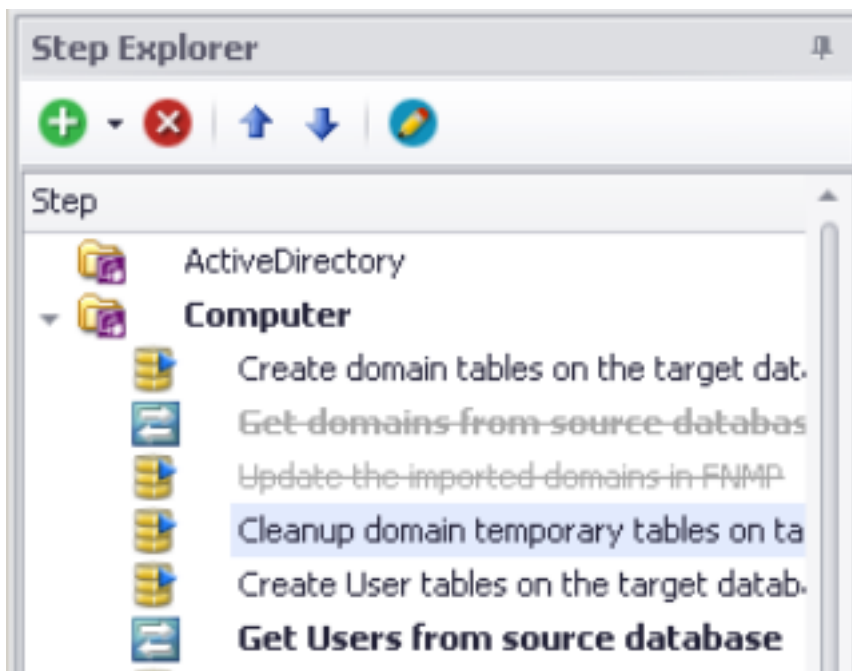
Edit panel

The edit panel consists of three main areas:

• A properties section

- An SQL script

- A log message and result panel.
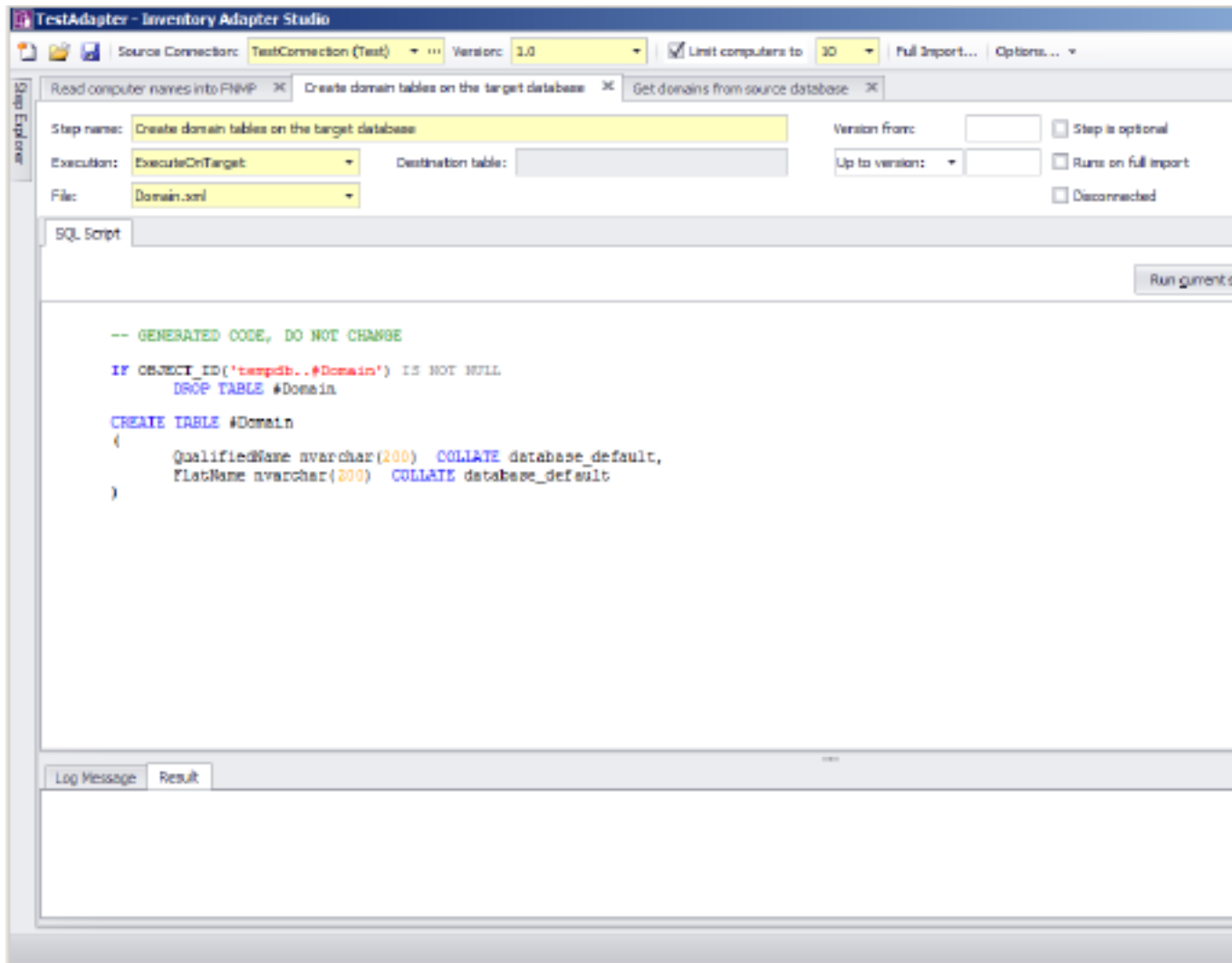
Properties section

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.

Value	Supported modes	Step type	Notes
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`.
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.
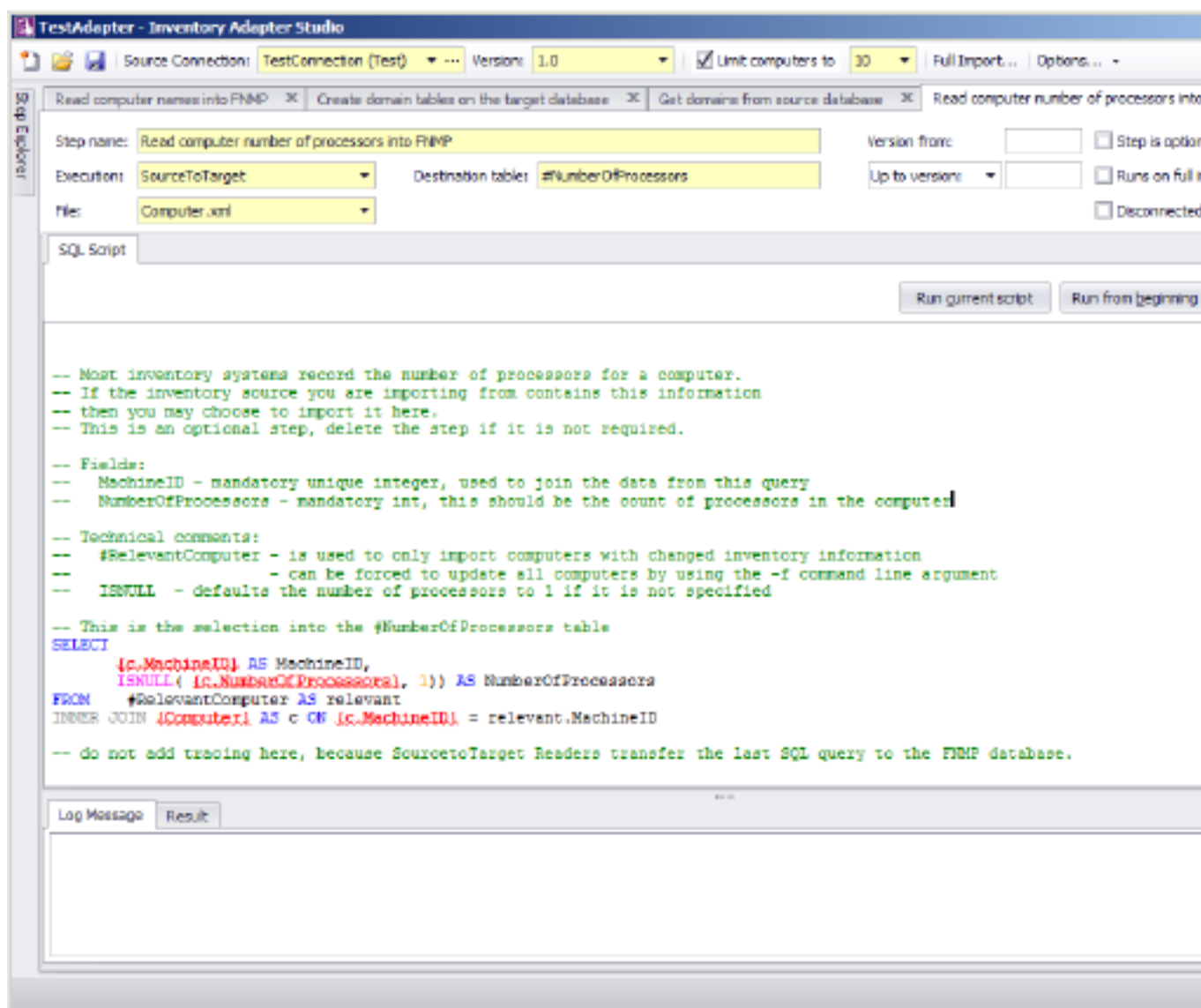
Figure 2: Red underlined text should be replaced with your own database column and table names.

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.
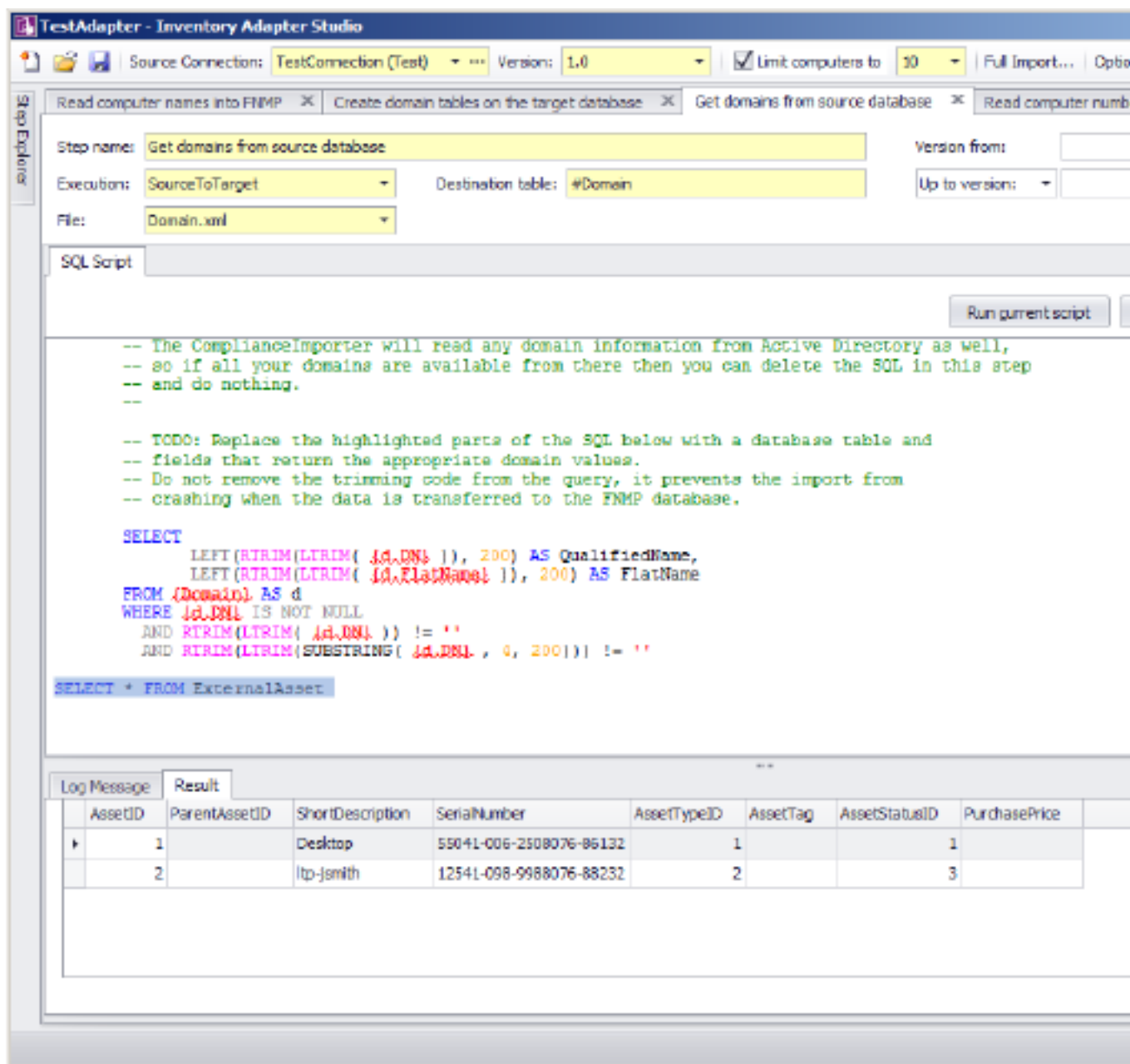
Installing Inventory Adapter Studio

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-premise 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 Software** > **Inventory Adapter Studio**).

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* on page 213).

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 fulfil 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* on page 202.

- "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 Software 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 6: 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.	Disconnected

Type of operation	Description	Available in modes
	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 on page 213).	
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.
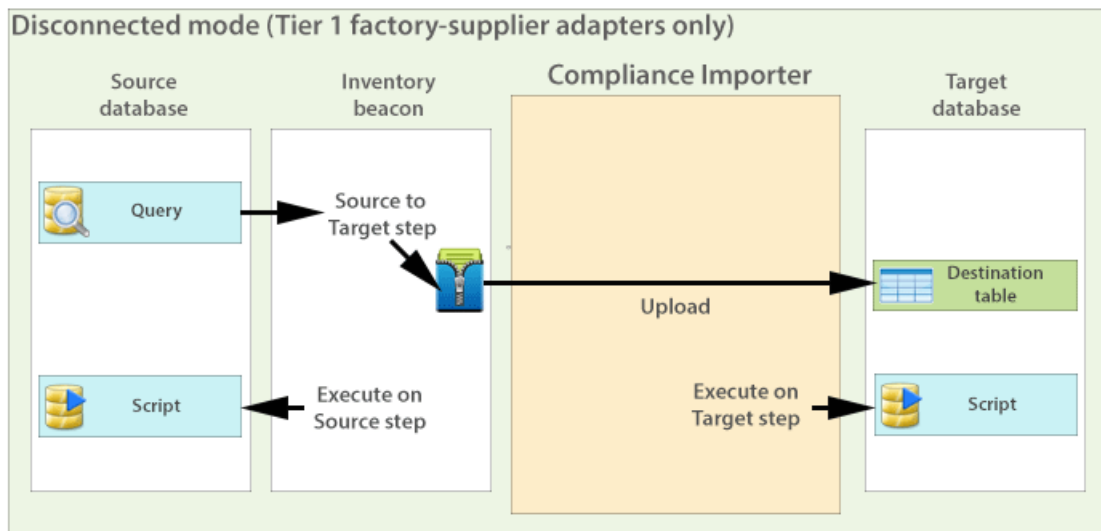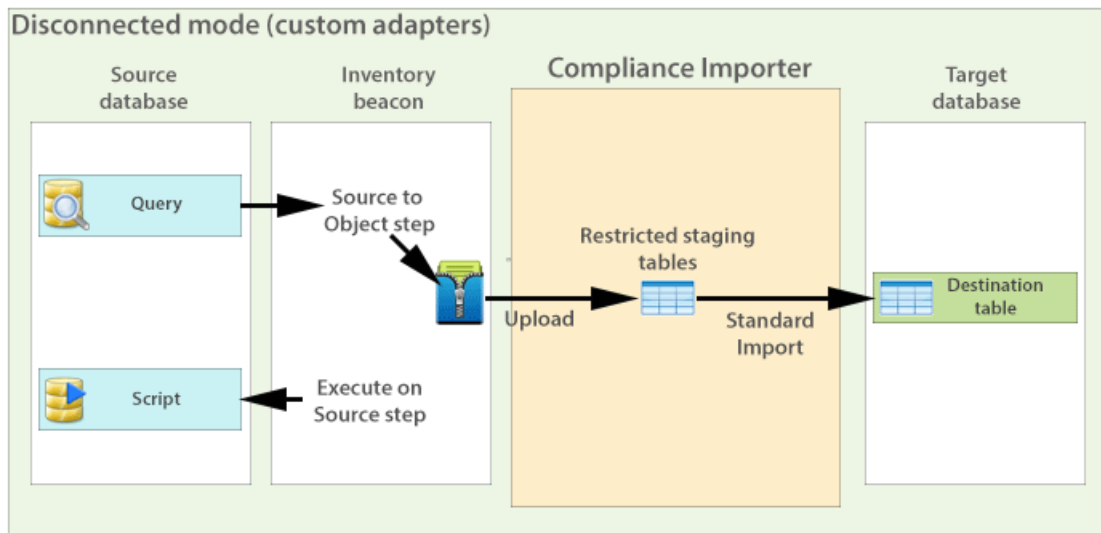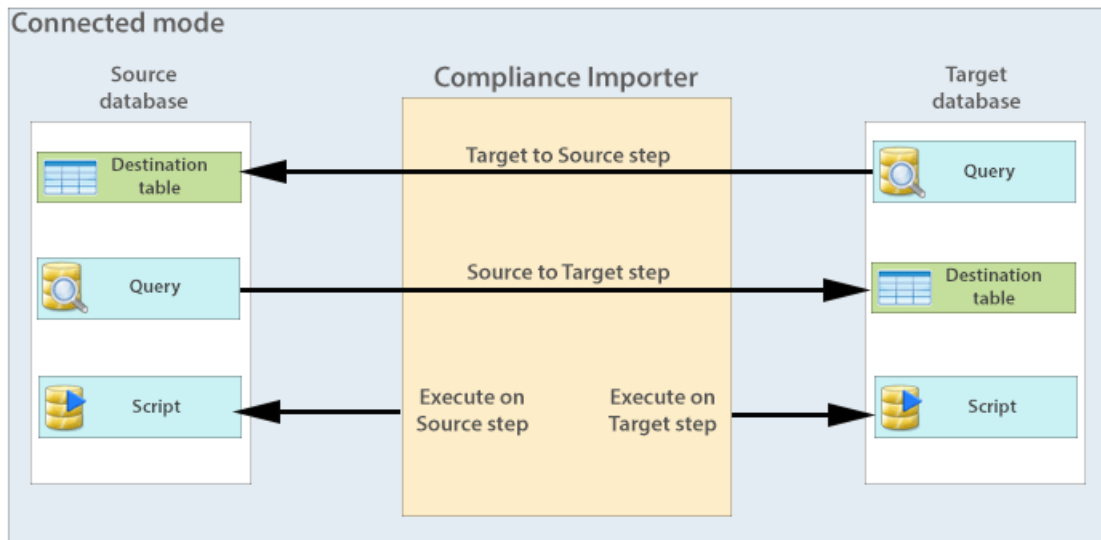
Structure of Templates for Inventory Adapters

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.

To Create a New Adapter

Use this process to create an adapter from scratch. There is a separate process for editing an existing 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.

1. Click the New icon in the toolbar.

    The **Create New Adapter** dialog opens.
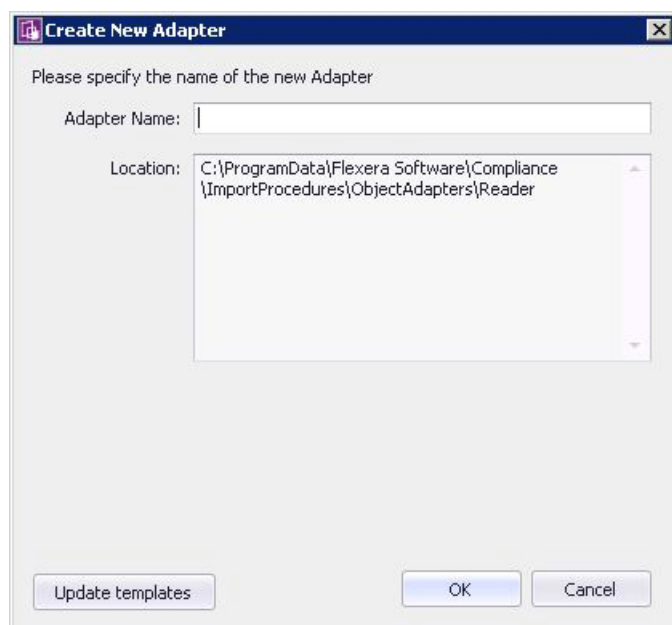
Tip • *If the Inventory Adapter Studio cannot locate downloaded templates, it displays a warning message in this dialog. You can download the latest templates using the* **Update templates** *button (if necessary, first re-establishing your link to the application server in the inventory beacon interface).*

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.

To Edit an Existing Adapter or Template

You may edit an existing, custom adapter that was created in your enterprise.

You may edit your custom adapters. 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.

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`.
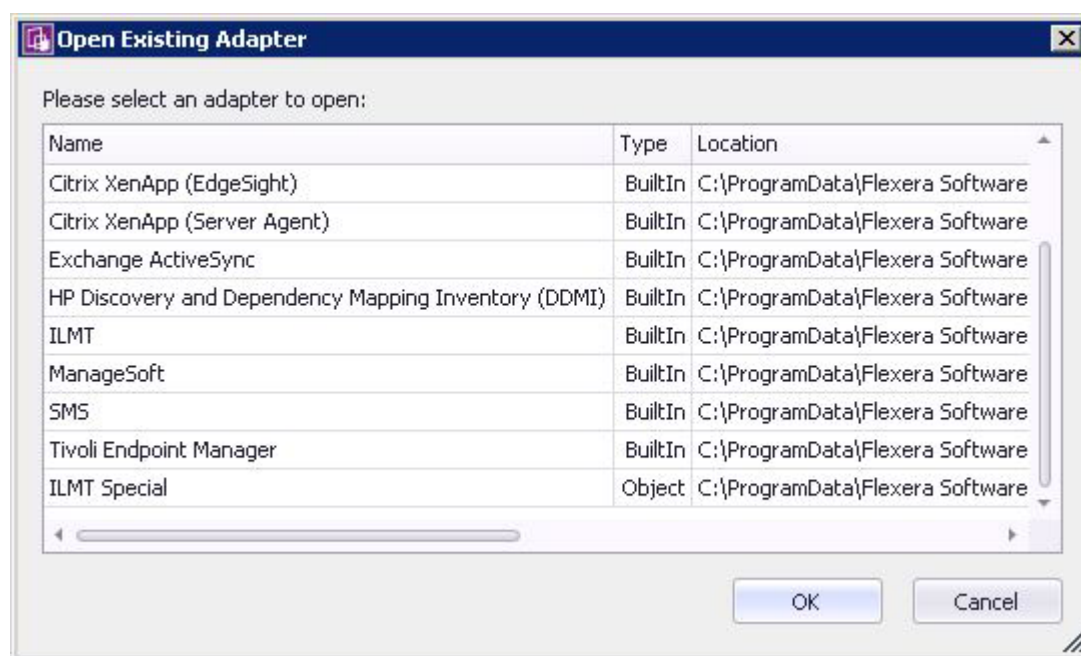
2. Select the desired custom adapter from the list, and click **OK**.

   Details of the adapter appear in the **Step Explorer** and edit panel.

To Create a Source Connection

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).

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 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.

1. In the toolbar, on right-hand end of the **Source Connection** control, click the **[...]** ellipsis button.
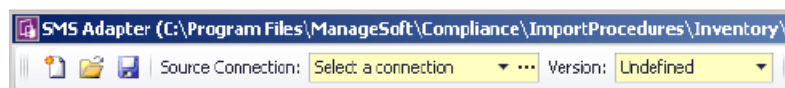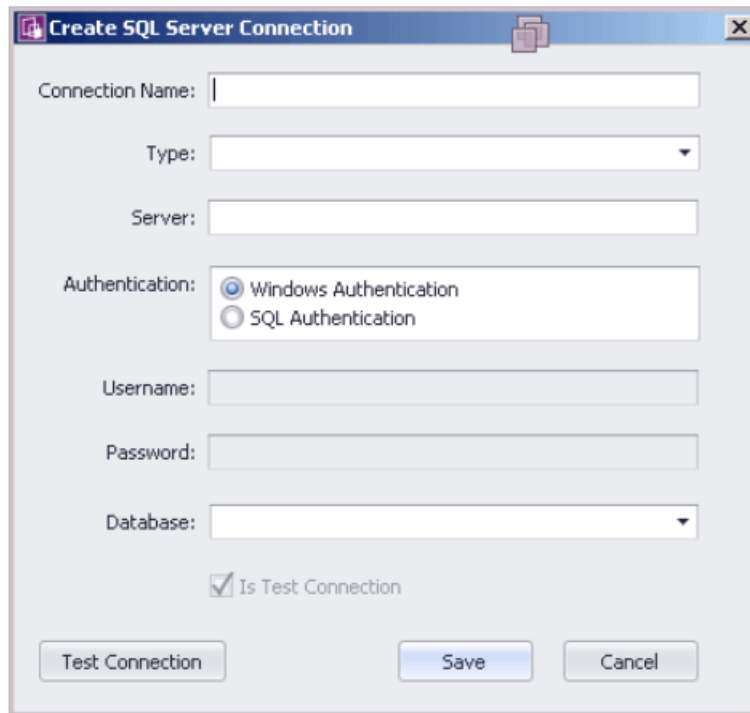
The **Select Source Connection** dialog opens.
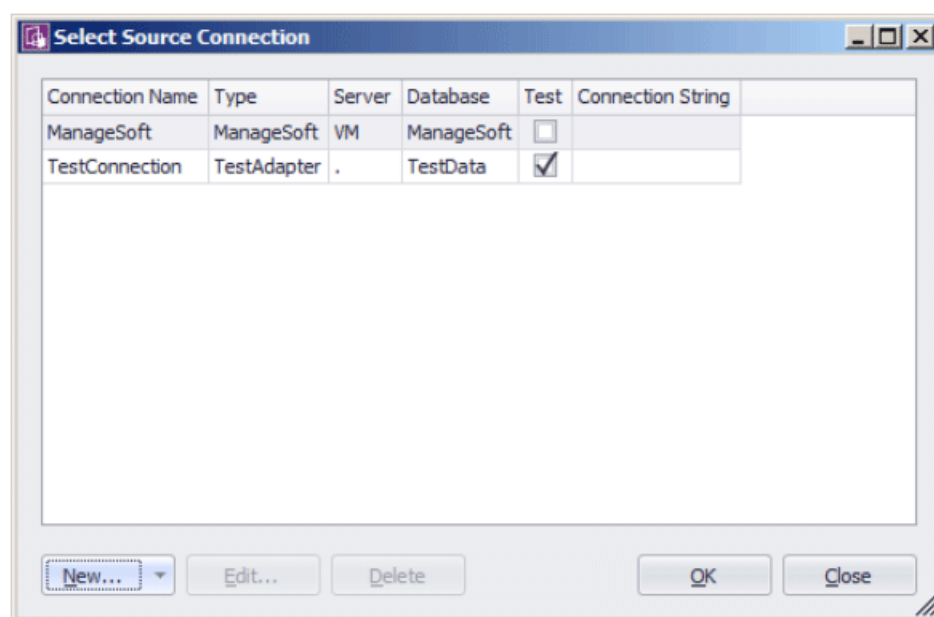
*Tip • You must create a new test connection for this adapter. You may not reuse an existing connection for this adapter. (The existing connections are those previous declared on your inventory beacon, and editing or deleting from this dialog affects the connections for your inventory adapter.) Only 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.*
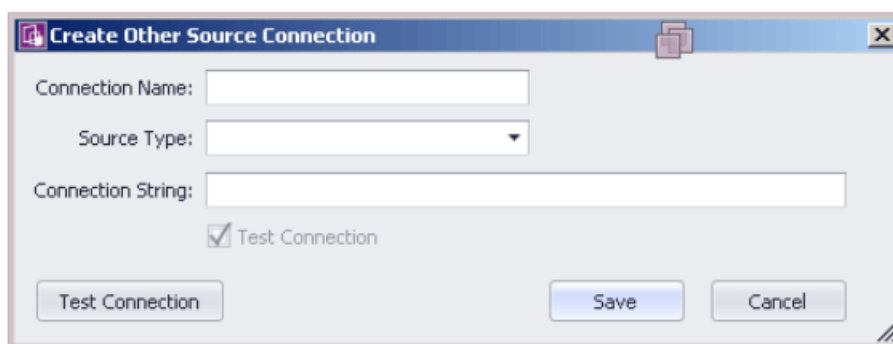
Overview: Process for Developing an Inventory Adapter

Here is your mental roadmap through the development process for inventory adapters, with links to the details.

1. Create a new adapter, normally pre-populated with an appropriate set of steps to gather and process data (*To Create a New Adapter* on page 195).

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* on page 197.)

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* on page 201)

   • Remove any steps provided automatically that are not required in your inventory adapter (see *Remove a Step from an Inventory Adapter* on page 202)

   • Change the execution order of steps within your inventory adapter (see *Reorder Steps in an Inventory Adapter* on page 202)

   • 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* on page 205 and *Testing an Adapter* on page 209). 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 page 205). 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 *To Publish Your Adapter* on page 212).

Adding a New Step to an Inventory Adapter

The add icon in the **Step Explorer** allows two broad kinds of additions to the current folder of steps.

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.)

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.
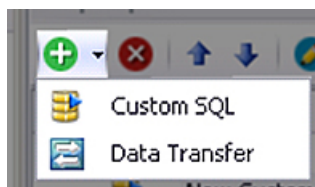
The new step appears in the **Step Explorer**, and its details appear in the editing pane on the right, potentially in a new tab (if you are already editing other steps).

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* on page 184.

Remove 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.

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).

Reorder Steps in an Inventory Adapter

You can arrange the steps within a folder in any order you prefer.

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.

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.

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.

To Select 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.

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.
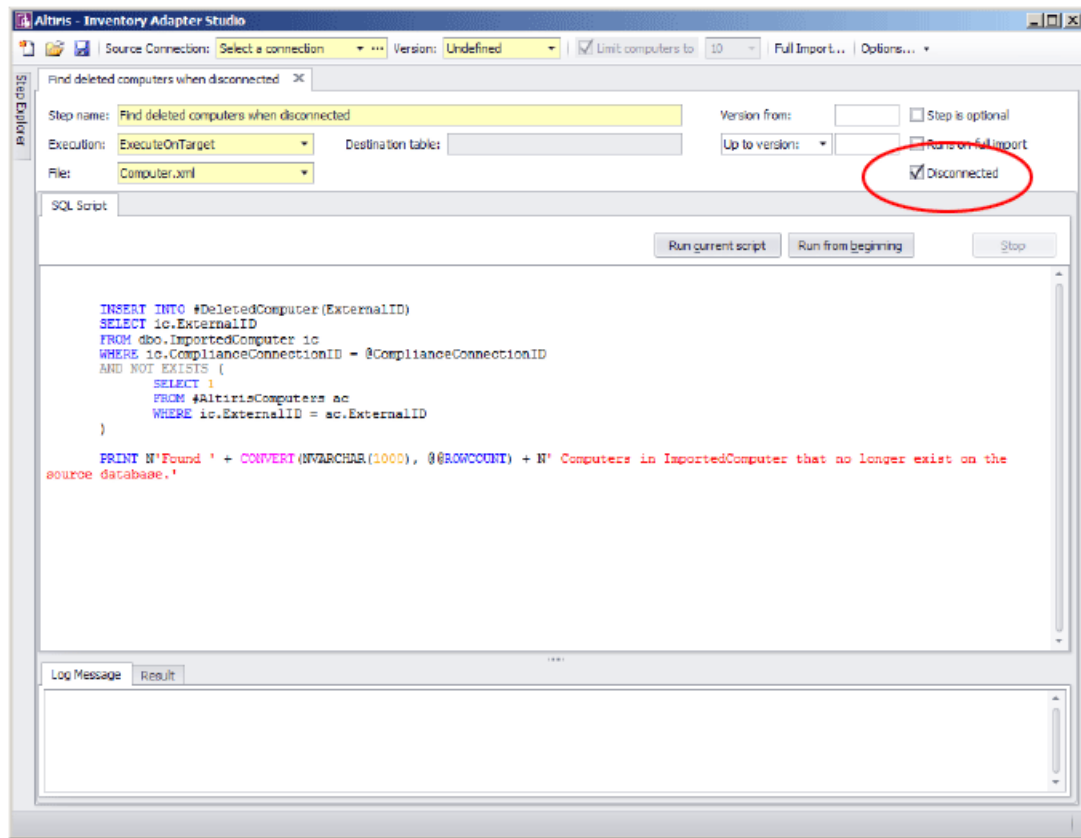
While this check box is selected, this step runs only when the adapter is executed in disconnected mode, such as on an inventory beacon. This setting is irrelevant when the **Execution** property is set to `TargettoSource` or `ExecuteOnTarget`, as all such steps are disabled for disconnected mode.

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`

Logical step described	Step type
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.

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* on page 184.

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.
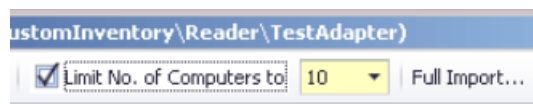
Start with a template

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.
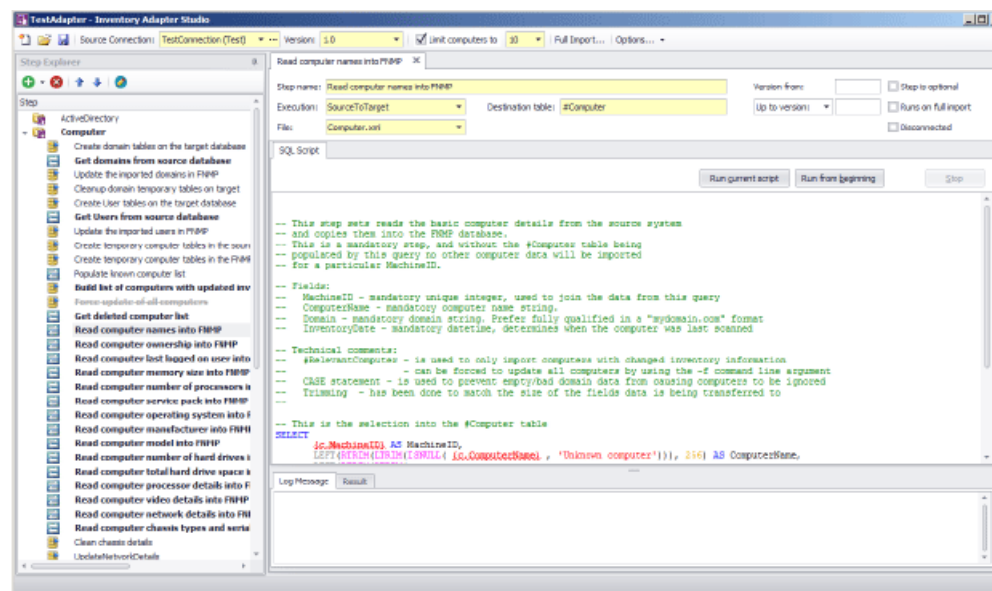
Test each SQL step

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.
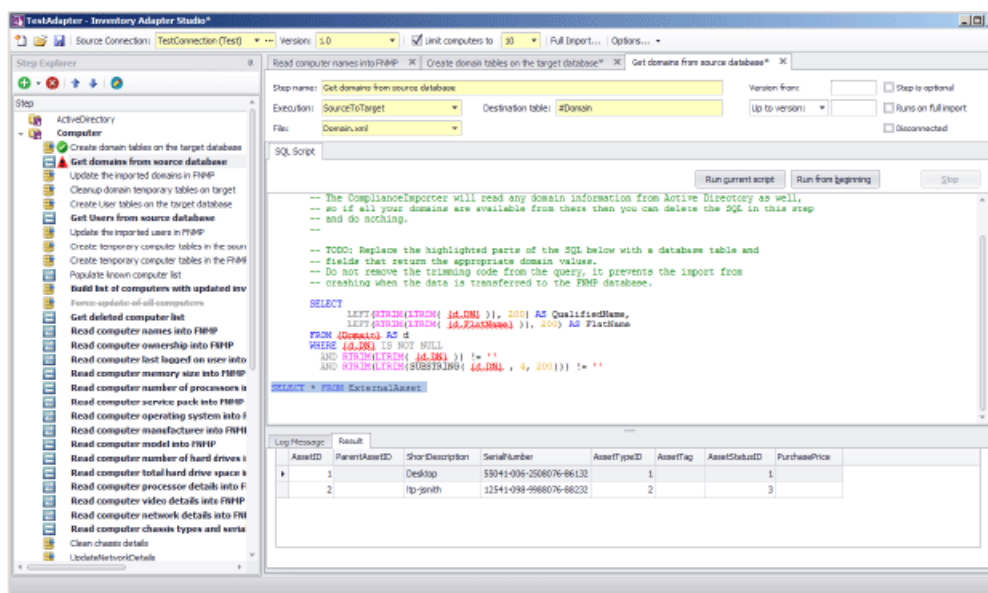
Figure 3: This step still requires customization, but the customizable text is not included in the selection, which can safely be run to inspect the results of the individual statement.

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.
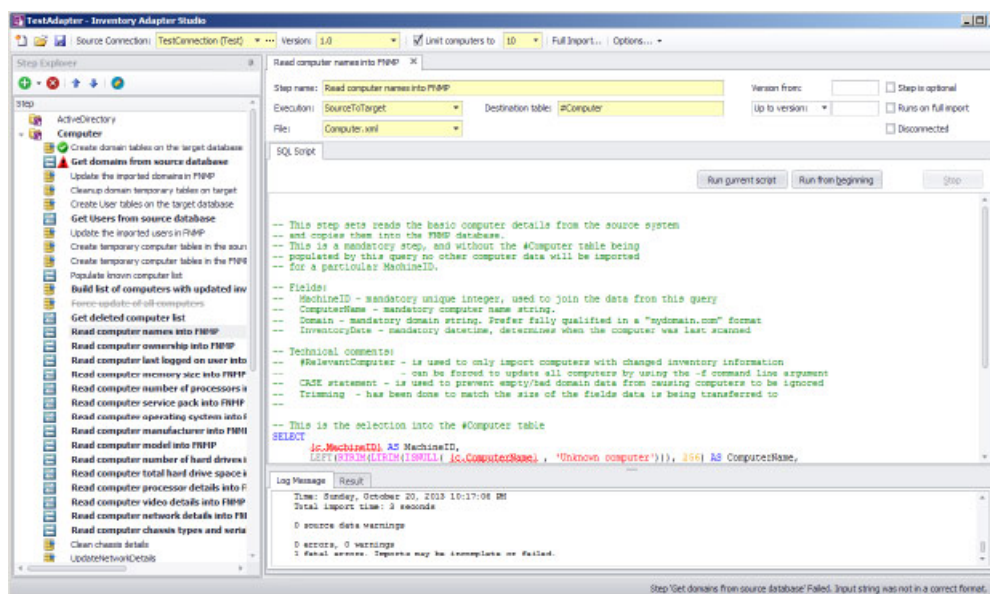
For repeated testing of an adapter collecting differential inventory, keep in mind that second and subsequent passes will not collect any results until there is a new inventory collected on a later date for some machine(s). In other words, as you continue testing on any given day, you can expect diminishing 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.
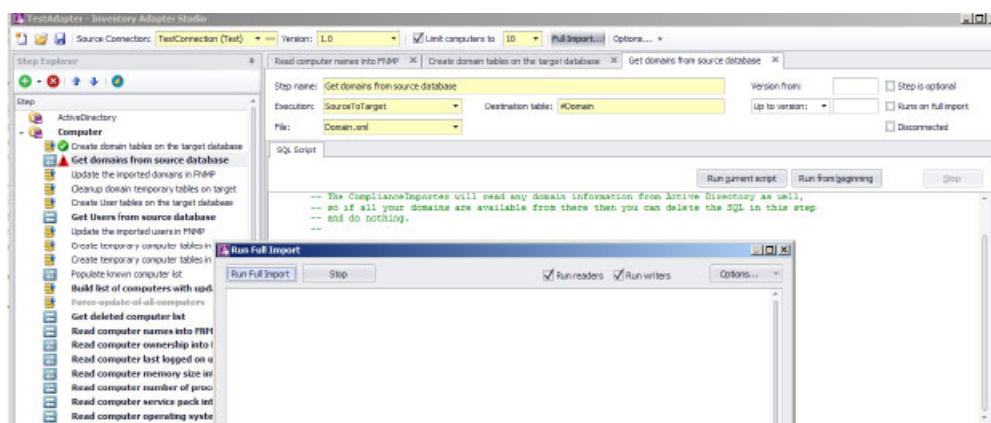
To Save an Adapter

Nothing unusual about this!

Save early, save often.

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.

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.

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.

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]
```

To Diagnose Writers for Your Adapter

Because your adapter runs on the central compliance server, you can inspect the database directly to see the results of your inventory import.

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 MMC user interface.

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) Right-click the **FlexNet Manager Platform** node in the console tree, and select **Compliance import...** from the context menu.

   b) Ensure that, to calculate compliance with your newly-imported data, you have selected the **Inventory import** radio button.

   c) Click **Run Importer** to begin the compliance import process, and track progress in the log window.

2. When the reconciliation is complete, examine your imported information, for example in the following locations:

   • The **Hardware Assets** > **Inventory** node 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 **Users** node shows all the users you have imported and the attributes that have been set.

   • The **Software Assets** > **Evidence** > **Installers** node shows all the installation evidence that was imported.

   • The **Software Assets** > **Evidence** > **Files** node shows all the file evidence that was imported.

   • The **Software Assets** > **Applications** node 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 is displayed on licenses, but for this validation you need to see it on applications as well). The following query will show 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
```

To Publish 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.

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.
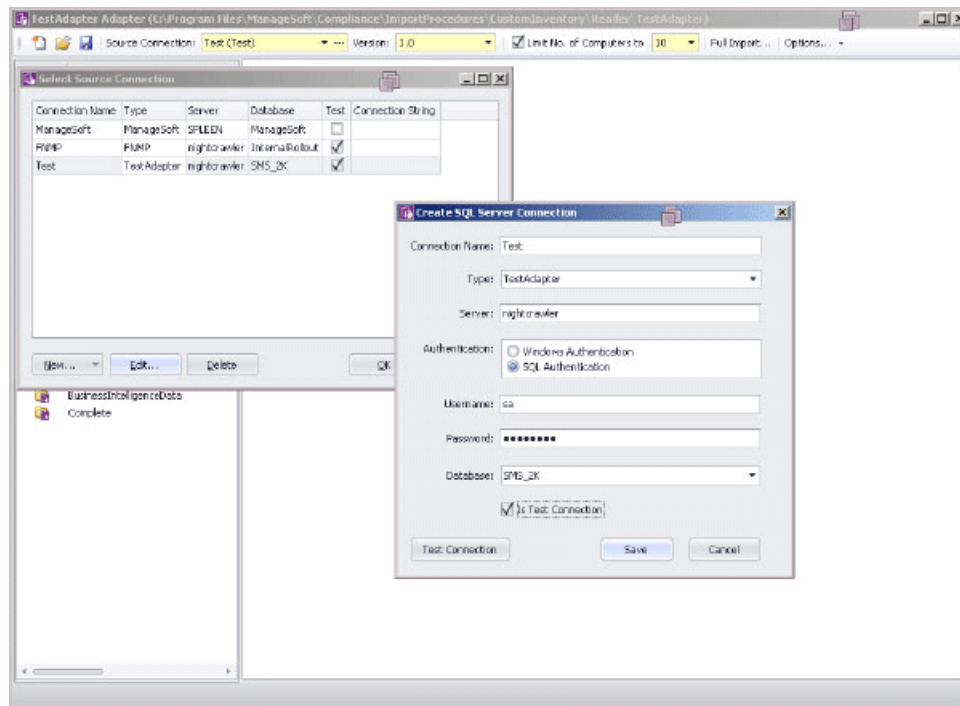
4. Click **Save**.

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.

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: ActiveSyncDevice

These properties of ActiveSyncDevices may be uploaded by inventory adapters.

`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.*
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.	The IMEI of the device.
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.

Property	Attributes	Notes
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: Cluster

These properties of Clusters may be uploaded by inventory adapters.

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. Values may be: • 1 = vMotion Cluster • 2 = Hyper-V Cluster • 3 = Host Affinity Group • 4 = VM Affinity Group.
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.
InventoryAgent	Alpha-numeric text (maximum 64 characters). Default: ''. May be null.	The name of the person or tool that performed the last inventory.

Property	Attributes	Notes
`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

These properties of ClusterGroups may be uploaded by inventory adapters.

`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`).	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. Factory use only: DO NOT modify.
`ClusterTypeID`	A numeric reference into a static table. Default: 3.	Foreign key to the ClusterType table. Values may be: • 1 = vMotion Cluster • 2 = Hyper-V Cluster • 3 = Host Affinity Group • 4 = VM Affinity Group.
`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*

Property	Attributes	Notes
		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

These properties of ClusterGroupMembers may be uploaded by inventory adapters.

`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.*
`VCObjectID`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The identifier of the virtual machine in Virtual Center. *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*

Property	Attributes	Notes
		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

These properties of ClusterHostAffinityRules may be uploaded by inventory adapters.

`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.
`ClusterHostAffinityRuleType`	A numeric reference into a static table. Default: 1.	A unique identifier indicating a type of Cluster Host Affinity Rule. Values may be: • 1 = must run on • 2 = must not run on.
`ClusterHostGroupExternalID`	Unsigned integer (`bigint`).	The unique identifier for the imported cluster host group.
`ClusterVMGroupExternalID`	Unsigned integer (`bigint`).	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

These properties of ClusterNodes may be uploaded by inventory adapters.

`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. Values may be: • 1 = Active • 2 = Passive • 3 = Hot • 4 = Warm • 5 = Cold.
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

These properties of Computers may be uploaded by inventory adapters.

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.

Property	Attributes	Notes
ExternalID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the computer.

Property	Attributes	Notes
		  Note • Strictly, this attribute may be null, because it forms part of a compound database key. However, null keys 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 256 characters). May be null.	The name of the computer.
Domain	Alpha-numeric text (maximum 100 characters). May be null.	The domain of the computer.
OperatingSystem	Alpha-numeric text (maximum 128 characters). May be null.	The operating system of the computer.
ServicePack	Alpha-numeric text (maximum 128 characters). May be null.	The service pack installed for the operating system.
NumberOfProcessors	Unsigned integer (int). May be null.	The number of processors in the computer.
ProcessorType	Alpha-numeric text (maximum 256 characters). May be null.	The type of processor in the computer.
MaxClockSpeed	Unsigned integer (int). May be null.	The maximum clock speed of the fastest processor in the computer.
NumberOfCores	Unsigned integer (int). May be null.	The number of cores in the computer.
TotalMemory	Unsigned integer (bigint). May be null.	The total RAM in the computer.
ChassisType	Alpha-numeric text (maximum 128 characters). May be null.	The type of case of the computer.
NumberOfHardDrives	Unsigned integer (int). May be null.	The number of hard drives in the computer.
TotalDiskSpace	Unsigned integer (bigint). May be null.	The total size of all hard drives in the computer.
NumberOfNetworkCards	Unsigned integer (int). May be null.	The number of network cards in the computer.
NumberOfDisplayAdapters	Unsigned integer (int). May be null.	The number of graphics cards in the computer.
IPAddress	Alpha-numeric text (maximum 256 characters). May be null.	The IP address of the computer.

Property	Attributes	Notes
MACAddress	Alpha-numeric text (maximum 256 characters). May be null.	The MAC address of the computer.
Manufacturer	Alpha-numeric text (maximum 128 characters). May be null.	The manufacturer of the computer.
ModelNo	Alpha-numeric text (maximum 128 characters). May be null.	The model number of the computer.
SerialNo	Alpha-numeric text (maximum 100 characters). May be null.	The serial number of the computer.
HostID	Alpha-numeric text (maximum 100 characters). May be null.	An identifier for the host of the computer (when the computer is a virtual machine).
LastLoggedOnUser	Alpha-numeric text (maximum 128 characters). May be null.	The domain/username of the user last logged onto the computer.
InventoryDate	Date/time field. May be null.	The date the computer last had inventory reported.
HardwareInventoryDate	Date/time field. May be null.	The date when the hardware was last reported.
ServicesInventoryDate	Date/time field. May be null.	The date when a service was last reported.
InventoryAgent	Alpha-numeric text (maximum 128 characters).	The name of the person or tool that performed the last inventory.
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.
NumberOfSockets	Unsigned integer (`int`). May be null.	The number of sockets in the computer.
PartialNumberOfProcessors	Fractional number (`float`). May be null.	The fractional processor count available to this 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.
ComplianceComputerTypeID	Unsigned integer (`int`). May be null.	If the data source knows that the computer is a VM or VM host, record that data here. If the value is not known, this should be left as NULL, to allow the type to be inferred. Unless there is a very good reason, this should not be set to 1 (Computer) as inference rules are helpful in this case.

Property	Attributes	Notes
ILMTAgentID	Unsigned integer (`bigint`). May be null.	Store the unique ID used by the ILMT agent on this device, if the inventory source is aware of this value.
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.
NumberOfLogicalProcessors	Unsigned integer (`int`). May be null.	The number of logical processors in the computer.
IsRemoteACLDevice	Boolean (0 or 1). Default: 0.	Used to determine if the current record is a remote ACL based device.
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.
LegacySerialNo	Alpha-numeric text (maximum 100 characters). May be null.	A previous serial number of this computer that can also be used for matching.
UUID	A universally unique identifier. May be null.	The BIOS UUID of the computer.
IMEI	Alpha-numeric text (maximum 256 characters). May be null.	The IMEI of the device.
PhoneNumber	Alpha-numeric text (maximum 128 characters). May be null.	The phone number of the device.
EmailAddress	Alpha-numeric text (maximum 256 characters). May be null.	The email address associated with the device.
CalculatedUser	Alpha-numeric text (maximum 128 characters). May be null.	The domain/username of the calculated user from the source database.
LastSuccessfulInventoryDate	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

Property	Attributes	Notes
		computer or the reader for this record is not using an incremental update model.

Inventory Object: ComputerCustomProperty

These properties of ComputerCustomPropertys may be uploaded by inventory adapters.

`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: Domain

These properties of Domains may be uploaded by inventory adapters.

`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.

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.
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*

Property	Attributes	Notes
		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.
FlatName	Alpha-numeric text (maximum 200 characters). May be null.	The flat name of the domain.

Inventory Object: EvidenceAttribute

These properties of EvidenceAttributes may be uploaded by inventory adapters.

`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*

Property	Attributes	Notes
		loss through over-writing. It is therefore best practice to treat this field as mandatory.

Inventory Object: FileEvidence

These properties of FileEvidences may be uploaded by inventory adapters.

`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.

Property	Attributes	Notes
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.
ProductVersion	Alpha-numeric text (maximum 200 characters). May be null.	The product version number in the file header.

Inventory Object: ILMTPVUCounts

These properties of ILMTPVUCounts may be uploaded by inventory adapters.

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 summarised 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.

Property	Attributes	Notes
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.
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

These properties of InstalledFileEvidences may be uploaded by inventory adapters.

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*

Property	Attributes	Notes
		loss through over-writing. It is therefore best practice to treat this field as mandatory.
ExternalFilePathID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the path of 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.*
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

These properties of InstalledFileEvidenceUsages may be uploaded by inventory adapters.

InstalledFileEvidenceUsage objects are uploaded to the ImportedInstalledFileEvidenceUsage table in the operations (inventory) database.

The ImportedInstalledFileEvidenceUsage table holds a record of 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.*
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 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.*

Property	Attributes	Notes
LastUsedDate	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The last used date of the file evidence, in the form YYYY/MM/DD.
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, in the form YYYY/MM/DD.

Inventory Object: InstalledInstallerEvidence

These properties of InstalledInstallerEvidences may be uploaded by inventory adapters.

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, in the form YYYY/MM/DD.
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.*
ExternalInstallerEvidenceID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the installer evidence.

Property	Attributes	Notes
		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.*
`InstallDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The install date of the installer evidence, in the form YYYY/MM/DD.

Inventory Object: InstalledInstallerEvidenceAttribute

These properties of InstalledInstallerEvidenceAttributes may be uploaded by inventory adapters.

`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.
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.*
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.*
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.*

Property	Attributes	Notes
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

These properties of InstalledInstallerEvidenceUsages may be uploaded by inventory adapters.

`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). May be null.	The identifier used in the source connection for the user that the installer evidence was used on.
LastUsedDate	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The last used date of the installed installer evidence, in the form YYYY/MM/DD.
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, in the form YYYY/MM/DD.

Inventory Object: InstalledWMIEvidence

These properties of InstalledWMIEvidences may be uploaded by inventory adapters.

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.*

Inventory Object: InstallerEvidence

These properties of InstallerEvidences may be uploaded by inventory adapters.

`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.

Property	Attributes	Notes
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.*
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

These properties of InstallerEvidenceRepackageMappings may be uploaded by inventory adapters.

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.

Property	Attributes	Notes
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.*
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*

Property	Attributes	Notes
		loss through over-writing. It is therefore best practice to treat this field as mandatory.

Inventory Object: Instance

These properties of Instances may be uploaded by inventory adapters.

`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 CVS files collection date.
ExternalComputerID	Unsigned integer (`bigint`). May be null.	The identifier used in the source connection for the computer.
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`). May be null.	The identifier used in the source connection for the parent instance.

Inventory Object: InstanceUser

These properties of InstanceUsers may be uploaded by inventory adapters.

`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 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 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 user was created.
DefaultTablespace	Alpha-numeric text (maximum 256 characters). May be null.	The default tablespace for an Oracle user.
ExternalID	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the instance 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 user last logged on to the computer.
TempTablespace	Alpha-numeric text (maximum 256 characters). May be null.	The temporary tablespace for an Oracle user.

Inventory Object: LicenseUser

These properties of LicenseUsers may be uploaded by inventory adapters.

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 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 user, as reported in SAP. Does not necessarily map to a cost center in the GroupEx table.
Description	Alpha-numeric text (maximum 400 characters). May be null.	The description of the external user.
Email	Alpha-numeric text (maximum 400 characters). May be null.	An e-mail address for the external user.
EmployeeNumber	Alpha-numeric text (maximum 256 characters). May be null.	The employee number of the external user.
ExternalID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the external 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 user.
LastName	Alpha-numeric text (maximum 256 characters). May be null.	The last name or surname of the external user.
LicenseUserID	Unsigned integer (int). May be null.	The identifier of the external user in the LicenseUser 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.
UserName	Alpha-numeric text (maximum 400 characters). May be null.	The name of the external user.

Inventory Object: User

These properties of Users may be uploaded by inventory adapters.

`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 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 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 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 user, as reported in SAP. Does not necessarily map to a cost center in the GroupEx table.
Domain	Alpha-numeric text (maximum 100 characters). May be null.	The domain of the user.
Email	Alpha-numeric text (maximum 200 characters). May be null.	The email address of the user.
EmployeeNumber	Alpha-numeric text (maximum 128 characters). May be null.	The employee number of the user.
ExternalID	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the 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.*

Property	Attributes	Notes
FirstName	Alpha-numeric text (maximum 128 characters). May be null.	The first name of the user.
InventoryAgent	Alpha-numeric text (maximum 64 characters). May be null.	The name of the person or tool that performed the last inventory.
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 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 user.
SAMAccountName	Alpha-numeric text (maximum 64 characters). May be null.	The SAM account name of the user.
UserName	Alpha-numeric text (maximum 64 characters). May be null.	The account name of the user.

Inventory Object: VDI

These properties of VDIs may be uploaded by inventory adapters.

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
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.*

Property	Attributes	Notes
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.*
TemplateName	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	The VDI template 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.*
VDIGroupName	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	The VDI group 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*

Property	Attributes	Notes
		loss through over-writing. It is therefore best practice to treat this field as mandatory.

Inventory Object: VDITemplate

These properties of VDITemplates may be uploaded by inventory adapters.

`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*

Property	Attributes	Notes
		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

These properties of VDIUsers may be uploaded by inventory adapters.

`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 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 user has access to.

Inventory Object: VMPool

These properties of VMPools may be uploaded by inventory adapters.

`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.

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 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.

Property	Attributes	Notes
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.*
VCObjectID	Alpha-numeric text (maximum 256 characters). May be null.	The identifier of the virtual machine folder in Virtal Center.

Inventory Object: WMIEvidence

These properties of WMIEvidences may be uploaded by inventory adapters.

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*

Property	Attributes	Notes
		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.

7

The Business Adapter Studio

The Business Adapter Studio allows you to create and edit business adapters. These are ways of connecting to data sources in your enterprise and extracting relevant data for import into FlexNet Manager Suite.

This section introduces both business adapters, and the Business Adapter Studio that can use to custom-build them.

Introducing the Business Adapter Studio

What is a business adapter?

A business adapter is an XML file that:

• Defines a connection to a data source (which may be a database system within your enterprise infrastructure, or other things including a well-formed spreadsheet)

• Maps the columns from the data source to a standard set of objects and attributes that can be imported into the operations database for FlexNet Manager Suite.

Examples of business information that may be relevant to your software and hardware asset management include:

• Details of your organization structure (to form enterprise groups in FlexNet Manager Suite)

• Purchase orders (especially relating to software purchases, upgrades, and maintenance)

• Contract details

and the like.

How is a business adapter used?

Business adapters may be used in two modes:

• Connected mode where the adapter is run on your central application server with direct connection to your operations databases.

• Disconnected mode where the business adapter runs on an inventory beacon, and cannot directly access the operations databases. This mode requires tighter security, and regular uploads of archived business information are automatically uploaded to the application server, and processed in a separate stage that is not controlled by the business adapter.

In connected mode, a Windows scheduled task on the application server triggers the Business Importer to read the business adapter. In disconnected mode, running the adapter is separate from importing the results. Triggered by the inventory beacon on a daily schedule you specify, the business adapter is read by the Business Importer, which then

• Connects to the specified connection

• Gathers the data defined by the XML in the adapter

• Collects the results into an archive package on the inventory beacon in disconnected mode; or writes the data into staging tables in connected mode

• Immediately uploads the package to the operations database in disconnected mode; or processes the data from the staging tables into the operations databases in connected mode

• Repeats this process for each of the other currently enabled adapters awaiting execution.

Therefore in connected mode, the data is available in the web interface within a very short processing time; but disconnected mode takes a little longer. The uploaded packages are held in a staging directory until all

previous imports from business adapters are completed, and then the new arrival is processed into the operations database. Thereafter the business information is available in the web interface.

How is a business adapter created and maintained?

Business adapters are edited in the Business Adapter Studio. In this tool, you can:

- Develop the adapter in a protected, test environment, starting with templates that help keep your adapter compliant with the requirements of the central operations databases

- In connected mode (on your application server), simulate running the adapter to test and trouble-shoot its development

- Move the complete adapter into production.

Separately for business adapters running in disconnected mode, in the interface for the inventory beacon, you can turn any production-ready business adapter on or off (enabled/disabled), and schedule the time of day when all your enabled business adapters are run. As already noted, in connected mode, you use a Windows scheduled task to control timing.

Prerequisites

In brief:

- **System requirements:** Included in the requirements for the inventory beacon (for disconnected mode). The release notes for the inventory beacon are available from the download center on the Flexera Software website. The URL for this download location was provided in the e-mail you received from Flexera Software order processing, confirming your purchased products. The same website also contains the release notes for FlexNet Manager Suite, which include requirements for the Business Adapter Studio on the application server if you are preparing business adapters to run in connected mode.

- **Installation:** The process is different for the two modes (and the two installation locations). The Business Adapter Studio for disconnected mode is installed with the inventory beacon. It is expected to run on the same computer as the inventory beacon, and makes use of other services that the beacon provides. By contrast, for connected mode, the Business Adapter Studio must be installed separately on your application server, and instructions are included in the installation documentation.

Note • If you self-install the Business Adapter Studio on your application server, you must upgrade your installation to the latest version to take advantage of new properties introduced in the latest data model.

- **Skills:** Business Adapter Studio is intended for users comfortable with data models and mapping between them. It is an easy tool to use in that context, and provides guidance about available options. Templates are included that complete as much as possible of the definitions for you, and there are sample spreadsheets provided for those who prefer to standardize their datasets in that medium. You do not need SQL experience for disconnected mode adapters, as the Business Adapter Studio running on the inventory beacon does not allow for any custom SQL. In contrast, for connected mode, the Business Adapter Studio allows you to insert custom XML, but you should tackle this level of customization only if you are very confident that you will not cause disruption to your operational data (always use a test environment first!).

What is not suitable for a business adapter?

Business data does not include any inventory of hardware or software from your computer fleet. Inventory is imported during the inventory import process, for which a number of popular inventory tools are supported in a default implementation. You can also create inventory adapters to link non-standard inventory tools to the inventory import. You create inventory adapters using the separately available Inventory Adapter Studio, which is a separate tool from the Business Adapter Studio.
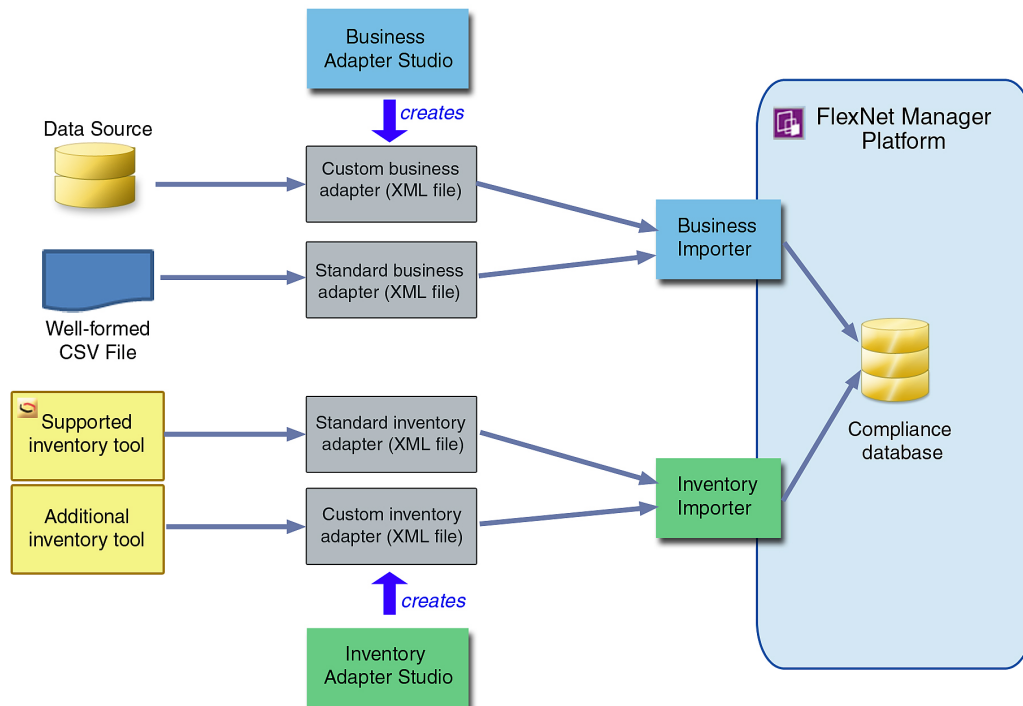
Overview: Development Process for Business Adapter

Business adapters import non-inventory data (such as purchases or enterprise structure) that helps to determine your compliance position.

There are two separate modes in which business adapters can operate:

- In connected mode, where they run on your central application server, and therefore have direct access to your central operations databases.

- In disconnected mode, where they cannot connect to the operations databases because they run on an *inventory beacon* that is remote from it. Here you have some restricted capabilities to increase security.

Important • In connected mode, data is injected directly into your operations database. You cannot see any evidence of the import in the Business Data tab of the System > Data Inputs page on your compliance console (this tab shows results only for imports from your inventory beacons).

Both these modes are outlined in the same list below. If an item does not specifically mention connected or disconnected modes, then it is applicable to both.

1. Launch Business Adapter Studio and add a framework for a new adapter (see *To Start the Business Adapter Studio* on page 254).

2. In connected mode (only), configure the adapter's connection to the FlexNet Manager Suite database. This is where imported data will be written by business adapters running in connected mode. See *Connecting to the Compliance Database (Connected Mode Only)* on page 273.

3. Configure the connection to the data source. This is where data will be imported from. See *Connecting to a Data Source* on page 258.

4. Confirm that you are querying the correct data from the data source. See *Reviewing Data from the Source* on page 276.

5. Load the list of properties from the data source, so that they can be mapped to objects in FlexNet Manager Suite. See *Retrieving the List of Fields* on page 277.

6. Link the objects in your source data to the objects that you want to update in FlexNet Manager Suite. See *Choosing Target Database Items in FlexNet Manager Suite* on page 279.

7. Define rules that manage updates and creation of these objects, based on the incoming data. See *Defining Import Rules for a Database Item* on page 281.

8. Define a mapping between the properties in the data source to those of the objects in FlexNet Manager Suite. See *Defining Import Rules for Attributes/Properties* on page 285.

9. In connected mode, you can simulate the data import to confirm it is working as expected. If necessary, you may use logging and tracing to debug any issues with the business adapter. Access all of these topics through *Testing and Diagnosis for Your Business Adapter* on page 295.

10. Save the adapter (*Saving Business Adapters* on page 257).

11. In connected mode, perform a real import from the data source into FlexNet Manager Suite (this calls the business importer to do the import). See *Running an Import from the Business Adapter Studio* on page 303.

12. Outside Business Adapter Studio, for connected mode only, configure a scheduled task to regularly run your new adapter. See *Setting Up Regular Imports (Connected Mode)* on page 306.

Managing Business Adapters

You can create, save, modify, and test adapters from within Business Adapter Studio. Access the Business Adapter Studio itself through your inventory beacon in disconnected mode, or through the Windows Start menu on your central application server.

While business adapters can be entirely managed within the Business Adapter Studio, they may also be deleted in Windows Explorer. If you choose this approach, remember to modify any associated scheduled tasks as well.

To Start the Business Adapter Studio

For adapters that run on an inventory beacon, start the Business Adapter Studio from your inventory beacon interface. Alternatively, for those that will run on your central application server, start it from the Windows Start menu.

Disconnected mode is used whenever the business adapter runs on a separate computer that is not the operations server for an on-premises installation of FlexNet Manager Suite. Typical examples are:

• You are using the software as a service version, where the operations server is in the cloud

• You are using an on-premises solution, but the business adapter will run on an inventory beacon, perhaps because of network partitioning that separates the data gathering from your central server.

When the adapter runs on your central, on-premises server and has simultaneous access both to the business data and to the operations database for FlexNet Manager Suite, this is called connected mode. There are separate methods for starting Business Adapter Studio in connected and disconnected modes.

Note • *The account running the FlexNet Beacon interface requires administrator privileges. In particular, when running the Business Adapter Studio, the account must have write privileges to the registry on the server where it is executing. If this privilege is not available, and you select the encryption option in the Business Adapter Studio, the product will fail with the error* `The type initializer for 'Flexera.BusinessImport.BusinessImporterCryptgrapher' threw an exception.`

1. In disconnected mode, where the finished adapter will run on an inventory beacon:

   a) Select the **Business Importer** tab in the user interface for the inventory beacon.

   b) Optionally, if you think that new templates and reference files may be available since you started the inventory beacon user interface (UI), you may click **Download Configuration**.

   As the configuration files don't change often, and are checked for currency each time that you start the inventory beacon UI, this is necessary only in special circumstances.

   c) Click one of the following buttons:

      • Click **New...** if you are starting development of a new business adapter.

      • Click **Edit...** to modify one of your existing business adapters.

      Business Adapter Studio displays an initial dialog to collect details, and opens the appropriate business adapter in the editing environment.

2. In connected mode, where the business adapter will run on your on-premises operations server:

   a) From the Windows Start menu, open the **Flexera Software** program group.

   b) In that group, click **Business Adapter Studio**.

   Business Adapter Studio opens a blank window.

Creating a New Adapter

From the inventory beacon UI, you can start working on exactly one new business adapter at a time. Once the Business Adapter Studio is open, however, you can start other new adapters, and work on each one in its own tab.

When the Business Adapter Studio is already open:

1. Do either of the following:

   - Click the New icon ( ) in the tool bar.

   - From the **File** menu, click **New...**.

   A shell for a new adapter is created in its own tab within Business Adapter Studio, and given a default name in the structure outline on the left (you can rename the adapter at any time).

Starting instead from the inventory beacon user interface:

2. In the user interface for the inventory beacon, select the **Business Importer** tab.

3. Click **New...**.

   Business Adapter Studio displays an initial dialog to collect details.

4. Select the appropriate **Adapter template** from the option list.

   The adapter templates correspond to the objects in the operations databases that you are allowed to import in disconnected mode (through an inventory beacon). Each type allows you to import a fixed set of attributes for that object, and sometimes a small set of links to other related objects in the database.

5. Give the business adapter a useful name (**Adapter name**) that will assist you with future maintenance. The adapter name will also be referenced by the business importer.

6. Choose how the finished business adapter will execute on its business connection to the third-party system in **Execute as**:

   - Choose **Windows (current account)** if the connection will use the account that is then executing the FlexNet Beacon engine

   - Choose **Windows (specific account)** if the Business Importer should use a different account to make the connection to the other system, or file share, and so on.

   If you choose the latter, the **Username** and **Password** fields are enabled, where you can provide the credentials for the specific account.

7. Click **Save**.

   A shell for a new adapter is created in Business Adapter Studio, ready for you to identify the connection to be used to gather the information.

Editing an Existing Business Adapter

Different choices are available depending on whether the Business Adapter Studio is already open.

You may be reopening an adapter that you have been working on recently, or you may be updating a business adapter originally created with an earlier release of FlexNet Manager Suite. As certain objects and attributes may be deprecated over time, you may see various alerts.

Use this process when starting from the inventory beacon (when the Business Adapter Studio is not already open):

1. In the inventory beacon, select the **Business Importer** tab.

2. In the list of **Current scheduled imports**, select the row identifying the business adapter you want to edit.

3. Click **Edit...**, and the **Edit business connection** dialog appears. (For more information about completing this dialog, see *Creating a New Adapter* on page 255.)

Choose either of these options when the Business Adapter Studio is already open (in either mode, on the inventory beacon or the application server):

• Click the Open icon (📂) in the tool bar.

• From the **File** menu, click **Open...**.

If you open an old business adapter that contains deprecated content, a large warning dialog appears, and the status bar displays the list of deprecated objects or properties. Click on the status bar to display the list more conveniently. Deprecated objects and properties have a different icon in the tree list (on the left hand side), and also have a text explanation such as `(Deprecated property)` alongside the name.

Best practice is to check the adapter and delete deprecated objects and properties.

💡
_____

Tip • Business adapters containing deprecated objects and properties can still be executed, but the behavior may be unpredictable and you are at risk of a failed execution.

Renaming a Business Adapter

You can name (or rename) a business adapter at any time during the development process.

The starting conditions, and the UI presentation, vary:

• In disconnected mode (when the business adapter operates on an inventory beacon), you gave the business adapter a name as you created it. This adapter name is shown as the top-most node in the structure tree on the left.

• In connected mode (when the business adapter is running on your operations server in an on premises implementation), the business adapter received a default name as you created it. The adapter name is the second level of the structure tree. Consider renaming this as you start your editing process.

As the adapter name will be referenced by the business importer, you should ensure it has a useful name that will assist future maintenance.

1. In the structure tree on the left side of the user interface for Business Adapter Studio, locate the node for this business adapter (see comments above).

2. Right-click that business adapter node.

3. From the context menu, select **Rename**.

4. Overtype the current name with your preferred name for the adapter, finishing with the **Enter** key.

Your change is saved in memory until you choose to save the adapter.

Saving Business Adapters

While saving is as you'd expect, there are a few options and a restriction.

☞

Restriction • On an inventory beacon in disconnected mode, you must not change the folder in which business adapters are saved.

To save the business adapter you are working on now, do either of the following:

• From the **File** menu, choose **Save**.

• From the tool bar, click the Save icon (🖫).

To save all the business adapters currently open in Business Adapter Studio:

• From the **File** menu, choose **Save All**.

💡

Tip • In either of these processes when you are in connected mode, if a business adapter has not previously been saved and remains unnamed, you are offered the chance to name it while saving. The name you supply is shown in the tab for this business adapter, if you have multiple adapters open.

To change the XML file name, or (in connected mode only) save to a new location:

• From the **File** menu, choose **Save As**.

💡

Tip • If you are on your central application server and relocating an operational business adapter, but intend to continue its use from the new location, don't forget to check any existing scheduled task that may reference its old name/location, and then remove the old file.

Keep in mind that renaming the XML file is a separate thing from changing the operating name of the business adapter itself (for which, see *Renaming a Business Adapter* on page 256).

Closing Business Adapters (and the Business Adapter Studio)

Look in the **File** menu.

From the **File** menu of the Business Adapter Studio:

- Choose **Close** to close the business adapter you are currently editing (and continue using Business Adapter Studio).

- Choose **Close All** to close all open adapters.

- Choose **Exit** to close Business Adapter Studio, including closing any open adapters.

Defining Connections for a Business Adapter

Business adapters must connect to other business systems to extract information.

Every business adapter needs a connection defined to the external repository of business data, from which information will be read. This is its connection to the *source*.

Beyond that, connection requirements are different in connected mode and disconnected mode.

- In connected mode (where the business adapter is running on the application server), the business adapter also needs a second connection, to the operations databases (specifically the compliance database) where imported data will be written. This is its connection to the *target* database.

- In disconnected mode (where the business adapter is running on the inventory beacon), the business adapter has no access to the operations databases. Instead, business information is archived into packages and uploaded to the application server, where it is processed nightly and finally loaded into the operations databases (specifically the compliance database). This does not require any connection details. (If you have scaled up to a multi-server implementation, the upload is to the processing server.)

Use the same processes both for creating new connections, and for modifying connection details when you edit an existing adapter.

Connecting to a Data Source

You can prepare adapters for a wide variety of external data sources, and the details required depend on what kind of source is in use.

All the data source adapters start with these common details. Next you will select a different help page for details on completing this task, depending on the type of data source you are working with.

1. Ensure that the desired adapter is open in Business Adapter Studio.
2. In the structure tree on the left, select the node identifying your business adapter (in disconnected mode, this is the topmost node; in connected mode, it is at the second level).

   A page of properties for your business adapter is displayed.
3. From the **Type** option list, select one of the available options. Your choice changes the lower part the page, and each option is documented below.
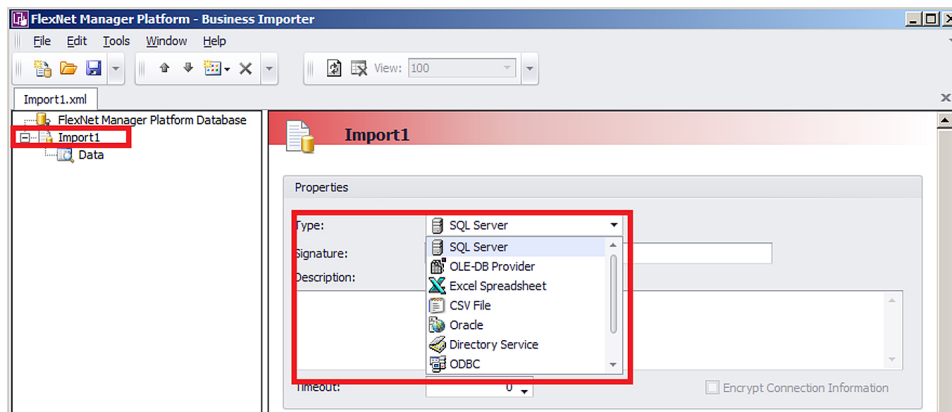
Figure 4: Choosing the Type for the data source

4. In the **Signature** field, enter the identity to be recorded against the change records generated in FlexNet Manager Suite against data using this adapter. If you leave the field blank, the default signature is `[USER NAME] ([IMPORT NAME])`. You may use any free-form text, and you may include either or both of these variables:

    *   `[IMPORT NAME]` — The name of the adapter

    *   `[USER NAME]` — The name of the Windows account under which the business importer runs, in the form `Domain\User`.

5. Enter a free-form **Description** for this business adapter. This may contain notes about the data source, notes about the adapter, reminders, and limitations.

6. In the **Timeout** field, enter the number of seconds to wait before giving up on a read request on the source data. The following values have special meaning:

    *   A value of `0` means there is no limit, and the business importer will wait indefinitely for the database read to finish.

    *   A value of `-1` means that the default time-out determined by the source database server should be used.

7. Select the **Encrypt Connection String** check box to encrypt the connection string details stored in the XML file for this adapter.

8. Depending on your choice for the **Type** option (above), the remaining panel on the page displays different content. The available choices are:

Option	Description
`SQL Server`	Connect with a Microsoft SQL Server database (see *Completing Connection Properties for Database Sources* on page 260)
`OLE-DB Provider`	Use for any data source that provides an OLE-DB compliant interface (see C*Completing Connection Properties for Database Sources* on page 260)
`Excel Spreadsheet`	See *Completing Connection Properties for Excel Spreadsheets* on page 262.

Option	Description
`CSV File`	Use for a file of comma-separated values (see *Completing Connection Properties for CSV Files* on page 264) 💡 *Tip • This type can also be used for importing general plain text files.*
`Oracle`	Use for an Oracle database (see *Completing Connection Properties for Database Sources* on page 260)
`Directory Service`	Use for an LDAP directory service such as Microsoft Active Directory (see *Completing Connection Properties for Directory Services* on page 270)
`ODBC`	Use this for a data source that provides an ODBC compliant interface (see *Completing Connection Properties for Database Sources* on page 260)
`Web Service`	Use for a SOAP web service (see *Completing Connection Properties for Web Services* on page 271)
`XML`	Use for an XML file (see *Completing Connection for XML Files* on page 273).

Completing Connection Properties for Database Sources

The SQL Server, OLE-DB Provider, Oracle, and ODBC sources share common input controls in the bottom panel of the adapter properties page.

1. To complete the **Connection String** field, click the ellipsis button (**...**) at the right end of the field.

   The standard Microsoft Windows **Data Link Properties** dialog appears.

2. Complete the required details, and the connection string is created for you.

   a) In **Select or enter a server name**, choose or enter a fully qualified server name (or IP address) for the server on which the database is running.

   b) Choose the authentication method for the account under which the business importer will access the database. **Use Windows NT Integrated Security** is recommended for easier maintenance over the long term; or you may use SQL authentication by choosing **Use a specific user name and password**, and entering the account details.

Important • *If you enter the account details, be certain to select **Allow saving password** so that the password can be brought into the adapter. Otherwise the password will be lost as soon as you click **OK**.*

c) From the **Select a database on the server** option list, choose the database.

d) Click **Test Connection** to make sure that your specifications are correct (adjusting as necessary for success).

e) Click **OK** to write these connection details for this adapter into Business Adapter Studio.

3. If you choose to edit the string or need more information, the following table provides additional notes for each database type.

Option	Description
`SQL Server`	• If you selected Windows authentication, a typical connection string (all on one line) is of the form `Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=SourceCatalogName; Data Source=SQLServerName` • For SQL authentication, a typical connection string is of the form `Password=SQLPassword;Persist Security Info=True;User ID=SQLAccount; Initial Catalog=SourceCatalogName;Data Source=SQLServerName`
`OLE-DB`	OLE-DB is a generic driver that can be used with any databases such as Microsoft Access, Ingres, Paradox, and others, provided that the corresponding OLE-DB driver has been installed and configured on the machine where the import is run. *Note •* *The business importer is a 32-bit application, and 32-bit OLE-DB connection strings must be used on 64-bit operating systems.* An example connection string for Microsoft Access: `Provider=Microsoft.Jet.OLEDB.4.0;Data Source=[Path and name of the .mdb file]`
`Oracle`	Oracle connections require the installation of an Oracle client provided by Oracle Corporation. The Oracle client installs the OLE-DB driver for Oracle. An example connection string:

Option	Description

	`Password=`*`Password`*`;User ID=`*`User`*`;Data`
	`Source=`*`OracleDataSourceName`*`;`
	`Persist Security Info=True`

ODBC	ODBC is a generic driver than can be used in conjunction with the Microsoft OLE-DB Driver for ODBC Drivers. The connection string varies according to the driver used. An example for a connection to an Excel file using a test DSN: DSN=test;DriverId=790;FIL=excel 8.0;MaxBufferSize=2048;PageTimeout=5;

4. Enter an SQL query in the **Query Text** field.

   This query runs against the data source, and return a grid of data that the business importer brings into the compliance database in FlexNet Manager Suite. Most of the rest of the Business Adapter Studio UI focuses on mapping the data returned by this query to the appropriate properties in the compliance database.

5. In the **Timeout** field, enter the number of seconds to wait before giving up a query from the data source. The following values have special meaning:

   • A value of `0` means there is no limit and the business importer will wait indefinitely for the query to finish.

   • A value of `-1` means that the default time-out determined by the source database server should be used.

Completing Connection Properties for Excel Spreadsheets

Complete these settings when the source business data is in a well-formed Excel spreadsheet.

As well as making these settings in the Business Adapter Studio, consider the registry settings documented below.

The following properties can be set when importing from an Excel spreadsheet.

Property	Notes
File name	The full path to the Excel file to import. 💡 *Tip • Click the ellipsis button (...) at the right of the text box to use Windows Explorer to locate the file.*
Worksheet	The specific worksheet to import from within the spreadsheet. Only one worksheet can be imported using each adapter.

Property	Notes
Auto-generate SQL Query	Selecting this check box causes Business Adapter Studio to automatically generate the SQL query for the worksheet (recommended). Alternatively, you may clear this check box and manually specify the query.
Query	The query to run against the Excel spreadsheet and extract data from the chosen worksheet.
Read "Intermixed" data columns as text	Selecting this check box causes the OLE-DB driver to resolve ambiguous columns as text. The driver uses the first several rows (default 8) to determine the data type of each column, and favors numeric when confused. A numeric setting causes the import to fail for any records containing text in such a column. Clear this check box to rely on the OLE-DB driver to determine the column type.
First row contains column names	Select this check box if the first row in the spreadsheet is a header row with the names of the columns, rather than a data row. Conversely, clear the check box if the first row contains values that should be imported.

You may also want to consider adjusting the following registry entries on the computer where the business adapter runs:

* In disconnected mode, on the inventory beacon

* In connected mode, possibly on the application server.

These registry entries are found under the following registry key:

* For 32-bit operating systems:

    `[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Jet\4.0\Engines\Excel]`

* For 64-bit operating systems:

    `[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Jet\4.0\Engines\Excel]`

Entry	Notes
`TypeGuessRow`	The format for each column is automatically assigned, based on a sampling of 8 rows. This may generate issues in some scenarios. For instance, if the 8 first rows of specific column include only numeric values, the column will be considered as numeric when string values may exist in other rows below.
	Depending on the scenario, this may cause import errors or values may be discarded. One way to solve the problem is to change the number of rows considered by Excel.
	To modify this behavior, adjust the entry `[TypeGuessRow]`.
	This value defines the number of rows to read to determine the format of a column. A value of zero indicates the full Excel worksheet will be read; this value may impact performances.

Entry	Notes
ImportMixedType	Depending on the quality of the data and different scenarios that may occur, there may be mixed data types in the same column (for instance numeric and string). In this case, data should always be considered as a string. To ensure this occurs, set the [ImportMixedType] entry to [Text].
	Make sure that you also select the **Read "Intermixed" data columns as text** field on the **Properties** page for the business adapter, as described above.

Tip • *If your Excel file to import includes multiple worksheets, the Business Adapter Studio needs further assistance with your import. You can take either of the following approaches:*

- *Make a copy of the Excel file and remove all the worksheets except the one you wish to import.*

- *Leave your spreadsheet unchanged; and modify the Business Adapter Studio configuration to control how this import is processed.*

To make this configuration change:

1. *In the Business Adapter Studio, from the **Tools** menu, choose **Options**.*

2. *In the **Options** dialog, change the **Show advanced options** setting to* Yes*, and click **OK**.*

3. *In your adapter definition, set the option to use physical databases to true, and specify a name for your database staging table.*

This adds the following two attributes to your adapter definition in the XML file:

```
<Import
        Name="FromMultiWorksheets"

        ...
        UsePhysicalTables="True"

        DataTableName="MyTableName"
        ...
/>
```

Completing Connection Properties for CSV Files

As well as for CSV files, you can use these settings for plain text file imports.

Importing data into the FlexNet Manager Suite database from CSV (Comma-Separated Values) files (or text files) is probably one of the most reliable and simple ways of loading data.

The following properties can be set when importing from a file of comma-separated values.

Property	Notes
File name	This is the full path to the CSV file to import.

Property	Notes
	💡 *Tip • Click the ellipsis button (...) at the right of the text box to use Windows Explorer to locate the file.*
Column delimiter	Choose the option that matches how the columns are delimited in the CSV file. 💡 *Tip • You may enter any printable character as a custom delimiter.* If you choose `Fixed Length`, you must specify the column width in a `schema.ini` file, prepared using the **Tools** > **ODBC Data Source Administrator** menu option.
First row contains column names	Select this check box if the first row in the CSV file is a header row with the names of the values, rather than a data row. Conversely, clear the check box if the first row contains data that should be imported.
Skip the first *nn* Row(s)	Specify a number of data rows to ignore when importing the CSV file. If the first row is flagged as column names, it skips this number of rows after the header row. This is useful for ignoring introductory comments, as may happen (for example) with a Microsoft LMS report.

Even though CSV imports are usually simple and reliable, there are a number of advanced options for configuring your system to import from CSV files:

• There are registry settings you can set to improve the success of the import process (see below).

• In addition, if you set the **Column delimiter** to `Fixed Length` or want to specify any special treatment of columns in the CSV file, you need to create a `schema.ini` file in the same folder as the CSV file. The business importer will extract column information from the `schema.ini` file when importing data from the CVS file.

🔲

Important •

If your imported CSV file uses a delimiter other than the one specified in the Microsoft registry entry (even if the separator is a simple tab character), you must use a schema.ini file to over-ride the registry setting. If you neither change the registry nor use a schema.ini, and use a different delimiter, the import will fail. The registry setting is located at `HKLM\SOFTWARE\Wow6432Node\Microsoft\Jet\4.0\Engines\Text\Format.`

The following registry keys (on the computer where the business adapter runs) may be set for 32-bit systems in:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Jet\4.0\Engines\Text]
```

and for 64-bit systems in:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Jet\4.0\Engines\Text]
```

Setting	Comments
MaxScanRows	The format for each column is automatically assigned, based on a sampling of 25 rows. This may generate issues in some case scenarios. For instance, if the 25 first rows of specific column include only numeric values, the column will be considered as numeric when string values may exist in other rows below. Depending on the scenario, an error will be generated or values will be discarded. Also any data surrounded by the text delimiter (a double quote ["], will be considered as text. This setting defines the number of rows read to determine the format of a column. A value of zero indicates the full text file will be read; this value may impact performances.
Format	Set the delimiter for each field value (replacing the default value of a comma). Any delimit character is allowed, except for double quotation marks ("). A blank space may be used as the delimit character. If for some reason you cannot change the registry setting on this server, you can over-ride the registry setting with a line in a schema.ini file.

Using Schema.ini

Placing a schema.ini file in the source directory with the CSV file can control the import process.

Column names, data types, character sets, and data conversions may be specified for the business importer using a schema.ini file. This file contains the definition of the columns for any text files in the current directory, and overwrites all other settings, including Microsoft registry settings. Using the schema.ini file approach is useful, for example, when you need to define fixed length fields, or specify a custom delimter.

For example, if you need to use a delimiter different than the one specified in HKLM\SOFTWARE\Wow6432Node \Microsoft\Jet\4.0\Engines\Text\Format (or the equivalent key for 32-bit systems), but for any reason you cannot update that registry setting, you may over-ride the registry with a setting in schema.ini. For example, suppose that the registry setting is CSVDelimited, but your imported file uses a Tab character as the delimiter. Until you create an appropriate schema.ini, the import will fail, typically by crushing all your imported columns into one column in the Business Adapter Studio. To over-ride the registry setting for a particular import, create a schema.ini containing a line such as the following:

```
Format=TabDelimited
```

Microsoft Windows offers an easy way to generate a default schema.ini file based on the existing text files in a directory. This method can be used to generate the schema.ini file and to then adjust it.

1. To initiate the process, do one of the following:

   - On a 32-bit machine, access the Windows Control Panel and select **ODBC** from the icons in the control panel.

   - On a 64-bit machine, run the following command: C:\Windows\SysWOW64\Odbcad32.exe.

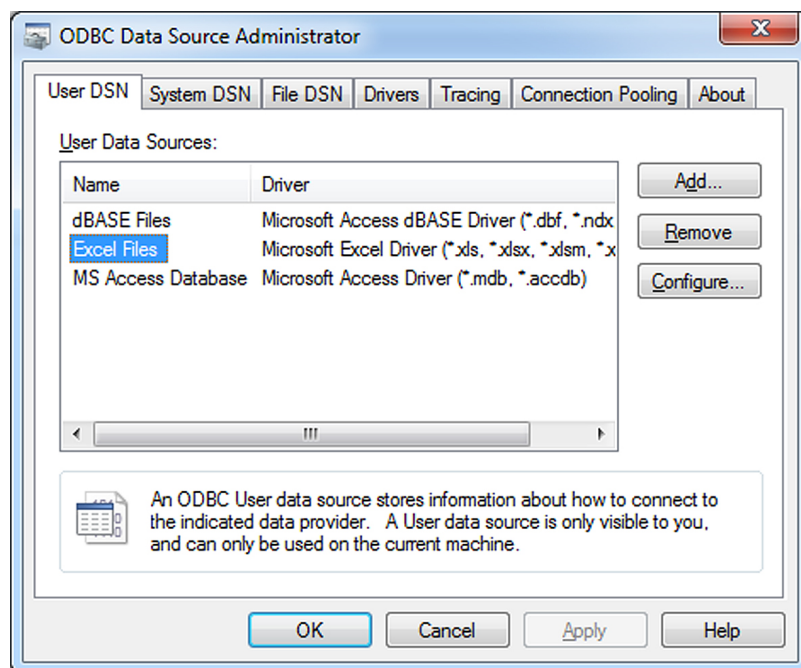   The **ODBC Data Source Administrator** properties are displayed.

Figure 5: The ODBC Data Source Administrator screen

Note • *This tool is primarily used to create and manage ODBC data sources. However, it is used here simply to create a default `schema.ini` file.*

2. Click **Add...**.

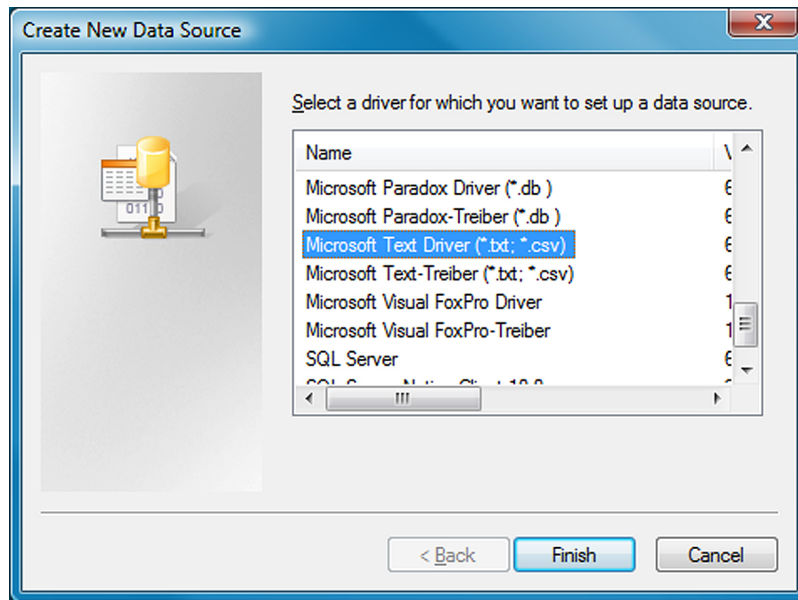The **Create New Data Source** dialog is displayed.

Figure 6: Pick the driver matching your data source

3. For CSV files or text files, select the **Microsoft Text Driver**.
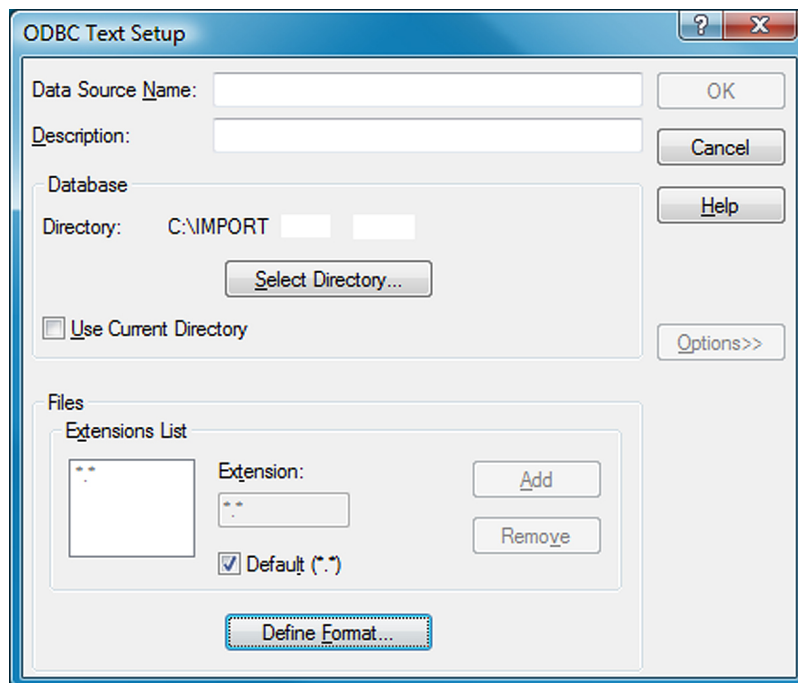
4. Click **Finish**.

   The **ODBC Text Setup** dialog is displayed.

Figure 7: Locate the CSV (or text) file

5. Click **Select Directory...** and browse to identify and select the CSV (or text) file that you want to use to import data.

6. Click **Define Format...**.

The **Define Text Format** dialog is displayed.
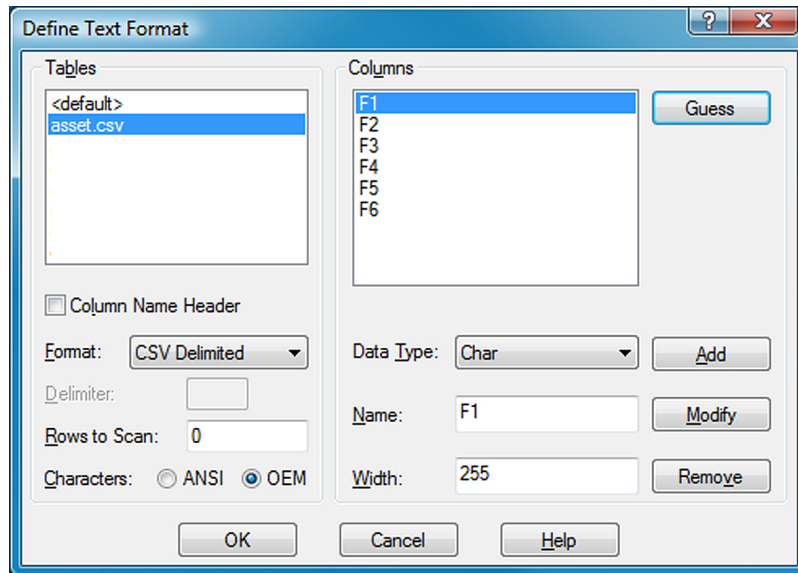
Figure 8: Define the format of the text file

7. Use this dialog to identify the columns of data in your text file, and the data type of each column.

Tip • *Click **Guess** to allow the program to analyze the text file and provide default table and column details. You can then modify any incorrect details.*

8. When you have finished defining the contents of the text file, click **OK** to return to the **ODBC Text Setup** dialog.

9. Click **Cancel**. The data source is not set up, but a new **schema.ini** file is created in the same folder as the text file. The `schema.ini` contains a definition of the tables and columns of the text file.

10. You can now edit the `schema.ini` file as desired, using a text editor such as `notepad.exe` or `wordpad.exe`.

Tip • *For further information about configuring a `schema.ini` file, see http://msdn.microsoft.com/en-us/library/ms709353(VS.85).aspx.*

Example for a CSV File Import

The `asset.csv` file located in the `temp` directory contains the following values:

```
Assetname, AssetSerialNumber,AssetPrice
    "First Computer", "SerialNumber1", 1000
    "Second Computer", "SerialNumber2", 2000
    "Third Computer", "SerialNumber3", 3000
```

The corresponding XML file used to load the assets in the repository would be:

```
<ImportName="ASSET"
        Type="CSV"
```

```
        ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;Data
  Source=c:\temp;Extended
  Properties='text;HDR=Yes;FMT=CSVDelimited'"
        Query="select * from [asset.csv]">
<LogName="NewLog"Output="file"Loglevel="debug"filename="[IMPORT
 NAME].log.txt"
    </Log>
<ObjectName="asset"Type="asset"Output="assetoutid"Update="True"Create="True">
    <Property
        Type="shortdescription"
        Name="Description"
        Value="AssetName"
        ValueType="FieldValue"
        UseForMatching="false">
    </Property>
    <Property
        Type="serialnumber"
        Name="Serial Number"
        Value="AssetSerialNumber"
        ValueType="FieldValue"
        UseForMatching="true">
    </Property>
    <Property
        Type="purchaseprice"
        Name="Purchase Price"
        Value="AssetPrice"
        ValueType="FixedValue"
        UseForMatching="false">
    </Property>
</Object>
</Import>
```

Completing Connection Properties for Directory Services

Context for the current task

The following properties can be set when importing from a directory service.

Property	Notes
Login	The account with which to connect to the directory service. *Note • Credentials are not required if the account is already a member of the target domain.*
Password	The password (if required) for the account connecting to the directory service.
LDAP PATH	The path to the LDAP directory entry. This is an empty string by default. The value of the path varies depending on the provider used.
Properties to load	Comma-separated list of properties to load from the LDAP directory.
Filter	Define a filter to restrict the number of rows returned from the specified properties. The filter is defined using the LDAP syntax, as customized by the vendor for the directory service. For example, Active Directory (ADSI) queries have the following requirements: • The string must be in parenthesis

Property	Notes
	• Expressions can use the relational operators `<, <=, =, >=, >` and the compound operators `&` and `|`. For example, the following filter returns all objects of category `user` and class `person` with a non-blank email address: `(&(objectCategory=user)(objectClass=person)(mail=*))`
Referal chasing	Defines how to handle referrals in the directory system. Possible values are: • `All` — Chase referrals of both subordinate and external types • `External` — (default value) Chase only external referrals • `None` — Never chase the referred-to server • `Subordonate` — Chase only referrals that are to a subordinate naming context in the directory tree.
Search scope	Sets the scope of the search. Possible values are: • `Base` — Limits the search to the base object, and only one object is returned • `One Level` — Search the immediate child objects of the base object, excluding the base object • `SubTree` — (default value) Search the whole subtree, including the base object and all its child objects.
Page size	An integer value (default 10,000) to set the number of records returned per page in a paged search.
Server page time limit	An integer value to limit the number of seconds that the server will search for an page result. The default value (`-1`) means to wait indefinitely.
Server time limit	Limits the number of seconds that the server spends on an entire search (including all pages). The default value of `-1` means that the server-determined default of 120 seconds will be enforced.
Size limit	An integer value to set the maximum number of objects the server will return in a search. The default value is `0`, which uses the server-determined default size of 1000 entries.
Client timeout	An integer value to set the maximum number of seconds that the client waits for the server to return results. The default value is `-1` which means to wait indefinitely.

Completing Connection Properties for Web Services

These additional details may be needed for connection to web services.

You need a detailed understanding of the web service that the business importer will connect to, using your adapter.

The following properties can be set when importing from a web service.

Properties	Notes
URL	This is HTTP URL of the web service.
Login	Account name with which to connect to the web service. 📓 *Note •* *Some web services do not require authentication, in which case you do not need to specify account and password.*
Password	The password (if required) for the account connecting to the web service.
Web service function or SOAP message	Set to one of: • The function name to call in the web service • The full SOAP request text. Function call example: `GetAllPurchaseOrders` results in the following SOAP request: ``` <?xmlversion="1.0"encoding="utf-8"?> <soap:Envelopexmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"xmlns:soap="http:// schemas.xmlsoap.org/soap/envelope/"> <soap:Body> <GetAllPurchaseOrdersxmlns=\"http://tempuri.org/" /> </soap:Body> </soap:Envelope> ``` Alternatively, if the SOAP request requires specific syntax or parameters, you can enter the full SOAP request. For example: ``` Query="<?xml version="1.0" encoding="utf-8"?> <soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema- instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http:// www.w3.org/2003/05/soap-envelope"> <soap12:Body> <GetAllPurchaseOrders xmlns="http://tempuri.org/" /> </soap12:Body> </soap12:Envelope>" ```
SOAP element	A string containing the name of the element to be read in the Web Service response. When the importer receives a response from the server, the full SOAP message is received, and the business importer may not know which of several XML elements contains the data to import. The business importer attempts to find elements in the following order (and reads data from the first one of these found):

Properties	Notes
	• The element you name in this field. • An element with a name made of the calling function name followed by the string "`Result`". In the example shown above, this would be `GetAllPurchaseOrderResult`. • The `<soap12:Body>` XML element.

SOAP header values	A string containing the values to be added to the Web Service request header. The values must be formatted as follows:
	
	`Name1=Value1;Name2=Value2;...`
	
	For example:
	
	`SOAPAction="http://MyServer/WebService/GetAllPurchaseOrders"`

Timeout	The number of seconds to wait before giving up on a a query from the data source. A value of `0` means there is no limit and the adapter will wait indefinitely for the query to finish. A value of `-1` means that the server-determined default time out should be used.

Completing Connection for XML Files

All you need is the path. And, of course, a well-formed XML file.

The following property can be set when importing from an XML file.

Properties	Notes
File name	The full path to the XML file to import. 💡 *Tip • Click the ellipsis button (...) at the right of the text box to use Windows Explorer to locate the file.*

Connecting to the Compliance Database (Connected Mode Only)

When you run your own application server, you can build business adapters that connected directly to your compliance database.

The connection to the compliance database is most conveniently made when Business Adapter Studio is installed on the same server as FlexNet Manager Suite (in this case you do not need to know any connection details). When the installations are on separate computers, it is helpful if the development computer has network access

to the compliance database. If not, you can work offline without an active connection to the compliance database, but the following limitations apply:

- You cannot load existing custom properties

- You cannot see values in lookup tables

- You cannot simulate data imports for testing, nor trigger real imports

- You cannot access the history of attempted data imports.

1. Ensure that the desired business adapter is open in Business Adapter Studio.

2. In the structure tree on the left, select the top node (**FlexNet Manager Suite Database**).

   A page for connection details is displayed.

3. Select one of the following options:

   - **Use the default settings from this computer** — Select this option only wherBusiness Adapter Studio is running on the same server as FlexNet Manager Suite. This option reads the connection details for the local compliance database from the registry key `SOFTWARE\ManageSoft Corp\ManageSoft \Reporter\CurrentVersion\DatabaseConnectionString`. If you have chosen this option, your definition is complete.

   - **Use the specific connection information below** — Select this option wherBusiness Adapter Studio is running remotely from the compliance database (but still in connected mode), or when you are configuring an adapter for an instance of business importer that runs remotely from the compliance database (but still in connected mode). This option requires you to complete the remaining steps.

   Important • Do NOT attempt to use this setting for business adapters that run on inventory beacons in disconnected mode.
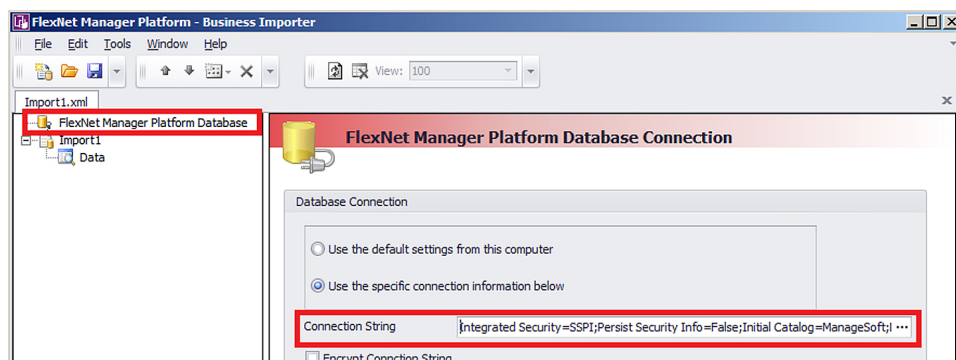
Figure 9: Defining the connection string to the compliance database

4. To complete the **Connection String** field, click the ellipsis button (**...**) at the right end of the field.

   The standard Microsoft Windows **Data Link Properties** dialog appears.

*Tip • Although using the **Data Link Properties** dialog is strongly recommended, it is possible to manually edit the connection string when required.*
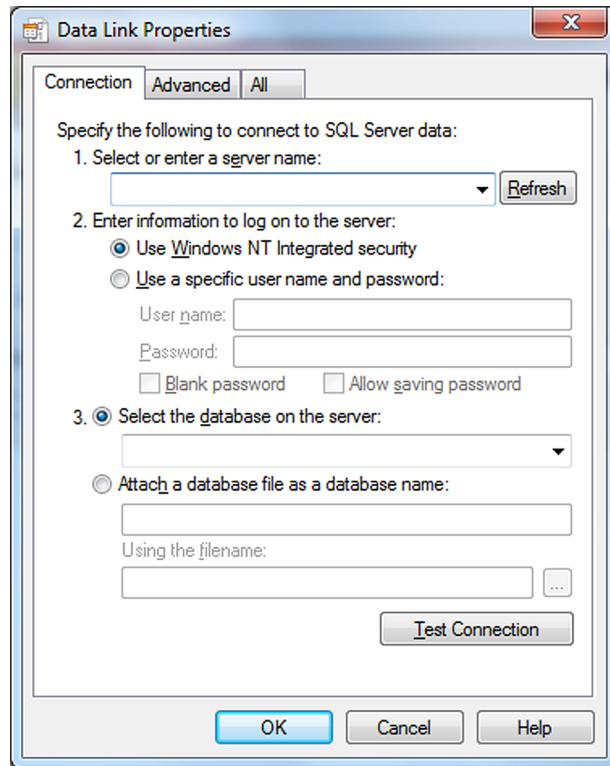

Figure 10: Specify connection to the compliance database

5. Complete the details in the **Data Link Properties** dialog:

a) In **Select or enter a server name**, choose or enter a fully qualified server name (or IP address) for the server on which the database is running.

b) Choose the authentication method for the account under which the business importer will access the database. **Use Windows NT Integrated Security** is recommended for easier maintenance over the long term; or you may use SQL authentication by choosing **Use a specific user name and password**, and entering the account details.

*Important • If you enter the account details, be certain to select **Allow saving password** so that the password can be brought into the adapter. Otherwise the password will be lost as soon as you click **OK**.*

c) From the **Select a database on the server** option list, choose the database.

d) Click **Test Connection** to make sure that your specifications are correct (adjusting as necessary for success).

e) Click **OK** to write these connection details for this adapter into Business Adapter Studio.

If you selected Windows authentication, a typical connection string is of the form `Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=ManageSoft;Data Source=SQLServerName`

For SQL authentication, a typical connection string is of the form `Password=SQLPassword;Persist Security Info=True;User ID=SQLAccount;Initial Catalog=ManageSoft;Data Source=SQLServerName`

Tip • A period (.) entered as the `Data Source` in the connection string uses the database connection on the current server where the business importer is running.

6. Select the **Encrypt Connection String** check box to encrypt the connection string details stored in the XML file for this adapter.

7. Save the adapter.

Reviewing Data from the Source

You can validate the returned data, with no effect on the source data or future imports.

Once the data source connection has been configured, the next step is to preview the data returned from the query. This confirms that you have configured the connection properly and are retrieving the expected data.

1. Click on the **Data** node in the structure tree on the left.

   The main panel displays the data returned from the data source.
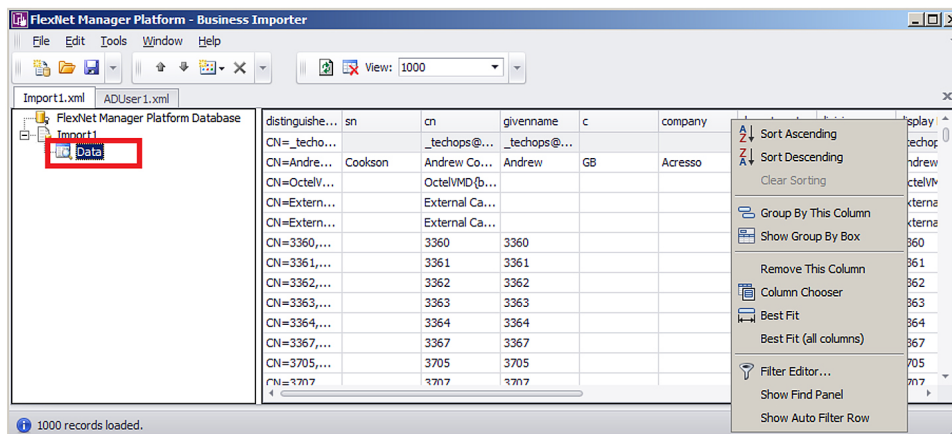
Figure 11: The data grid offers many options for organizing the list

2. Right-click on the column header to organize the data as you require.

Note • Operations on this list have no impact on the data source or data import. This list simply helps you confirm that you are gathering the correct data from the source.

Linking Data Imports to FlexNet Manager Suite

With the source data connection specified and checked, it's time to map the incoming source data to the destination fields within the operations databases (specifically the compliance database).

Mapping the source data to the destination database includes these steps:

1. Retrieving the list of fields from the data source (see *Retrieving the List of Fields* on page 277).

2. Choosing the target objects in the compliance database to which the data applies, and the order in which they should be populated in view of their interdependencies (see *Choosing Target Database Items in FlexNet Manager Suite* on page 279).

3. Defining the import rules to be applied to each imported item, for handling updates and object creation (see *Defining Import Rules for a Database Item* on page 281 for database objects, and *Defining Import Rules for Attributes/Properties* on page 285 for their individual attributes or properties).

4. Linking the source data fields, one by one, to the attributes (or properties) of the database objects in the target compliance database. For linking to objects, see *Defining Import Rules for a Database Item* on page 281. For their properties, see *Defining Import Rules for Attributes/Properties* on page 285.

Retrieving the List of Fields

Once confident that the correct information is returned from the data source, you may retrieve the field list from the data source. This step is required before you can start mapping data fields from the data source to objects in FlexNet Manager Suite.

1. From the structure tree on the left, select the name of this adapter (representing the XML file).

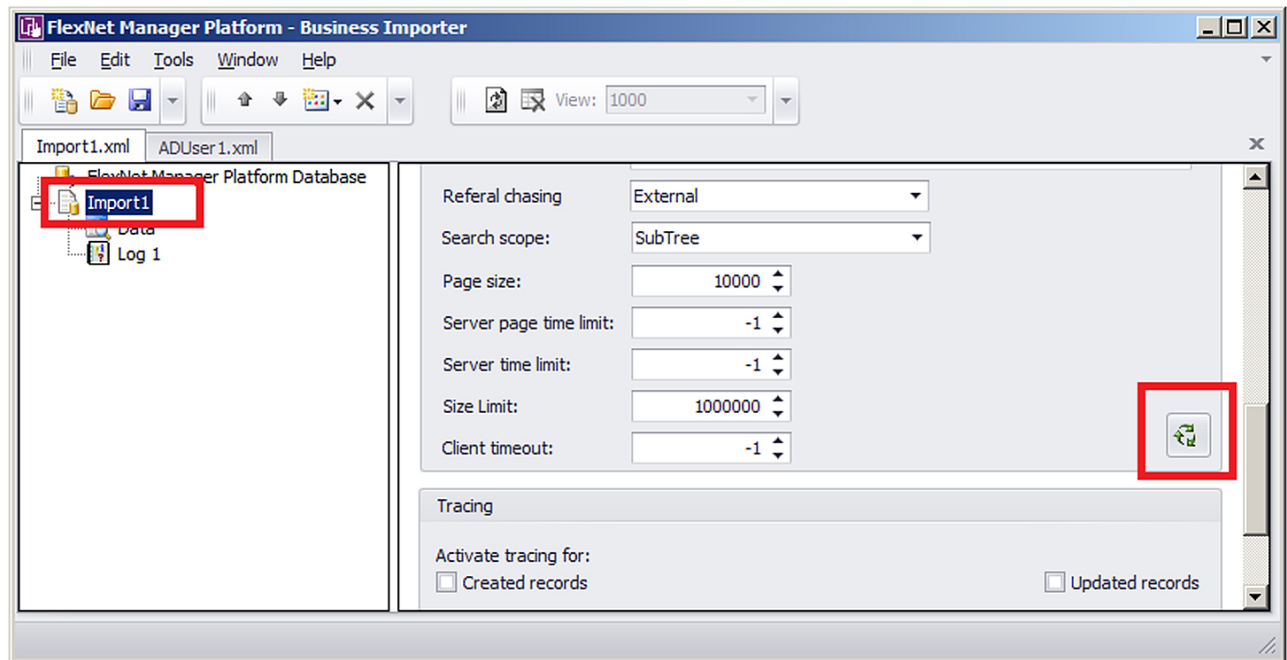2. In the data page, click the retrieval button ( ) on the right side.

Figure 12: The retrieval button fetches the field list into memory

The field list is fetched into memory. If there are problems, an error dialog appears; otherwise a success message appears in the status bar at the bottom of the user interface. The field list in memory is available for the next step, linking the imported data fields to the compliance database.

Updating Business Templates and Data Model

You can manually ensure an update of the local copy of the data model and templates used for business adapters (this is especially useful if you are adding custom properties).

The templates for business adapters, along with the sample spreadsheet files and the data model permissible for business adapters running in disconnected mode, are automatically updated daily (on the same schedule as inventory rules are downloaded to the beacon). In special circumstances, you may need these updated more immediately: for example, if you have just created a custom property on the application server, and want that custom property reflected in your business adapter, you can trigger an immediate download (when you don't want to wait through the rest of the 24 hour cycle).

Tip • The data model is updated before each download, so that it includes the latest data structure including custom properties.

It's better to update the templates and schema *before* adding the modified database object to your business adapter.

1. Ensure that the Business Adapter Studio interface is closed.

This permits the update of all downloaded files, and ensures that the new files are read when the Business Adapter Studio is reopened.

2. In the inventory beacon interface, select the **Business Importer** tab.

3. Click **Download Templates**.

4. Wait 2-3 minutes for the generation and download of all the data.

5. Still on the **Business Importer** tab, do one of the following:

   • Click **New...** to start a new business adapter using the latest templates and data structures

   • Select your preferred business adapter from the **Current scheduled imports**, and click **Edit...** to reopen the Business Adapter Studio and resume editing.

Tip • New properties are available only as you add the parent database object to your adapter. For example, suppose you already have a `Vendor` object in your business adapter, and you interrupt development to add a custom property to the `Vendor`. After completing the download process documented here, you need to delete your previously-entered `Vendor` object, and replace it with a new `Vendor` object so that you can access the custom property.

Choosing Target Database Items in FlexNet Manager Suite

The business adapter you are working on is open in the Business Adapter Studio.

Tip • If you are including custom properties in your business adapter that runs on an inventory beacon (in disconnected mode), make sure that, first:

• *The custom property has been defined on the application server*

• *You have downloaded the latest templates and data schema that includes your custom properties. For details, see Updating Business Templates and Data Model on page 278.*

1. Do either of the following:

   • Click the New Item button ( ) on the tool bar

   • Right-click the adapter node in the structure tree, and from the context menu select **Add New Item**.

   A context menu appears listing items from the compliance database. Many of these menu entries open sub-menus. (Menu entries vary in connected and disconnected modes.)
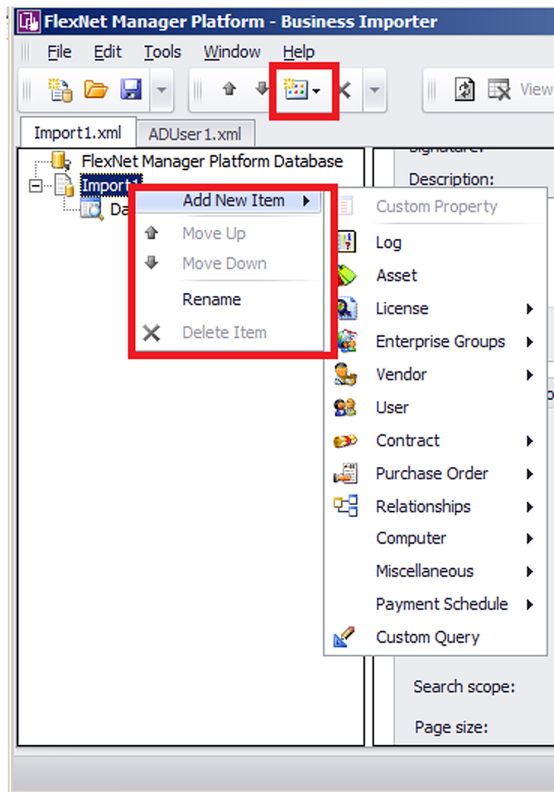
Figure 13: Choosing the target item in the compliance database

2. Select an item from the compliance database from the menu, working in logical order:

   • Select objects before any relationships they appear in.

   • Select objects before any other objects that refer to them. For example, if you have new vendor data and new purchase orders, ensure that the vendor object appears in the list before the purchase order object, as purchase orders refer to vendors.

   As you select an item, a new node is added to the structure tree under the adapter.

3. Repeat for all the compliance database objects needed.

4. If necessary, adjust the order of compliance database items by right-clicking an item and choosing **Move Up** or **Move Down** from the context menu. Remember that objects must receive imported data before they can be referenced by data imported to any other object.

Tip • *Expand a database item in the structure view to see the object's properties.*

Creating Import Rules

Import rules control creation and update of database items (in response to imported data) at two levels: object and attribute.

Once you have selected an item (object, relationship, or query) in the compliance database, you can establish the import rules for that item, including the source from your external data that should be loaded here. Import rules are available at two levels:

- The database objects themselves (such as vendor, purchase order, payment schedule), for which see *Defining Import Rules for a Database Item* on page 281

- The individual properties (or attributes) of those objects (such as name, telephone number, and so on), for which see *Defining Import Rules for Attributes/Properties* on page 285.

Defining Import Rules for a Database Item

Import rules may be set separately for objects and attributes. This topic covers the higher-level objects/items.

Your business adapter is open in Business Adapter Studio.

1. Select a compliance database item (object, relationship, or query) in the structure tree on the left side.

   The main page displays the import rules for that item.
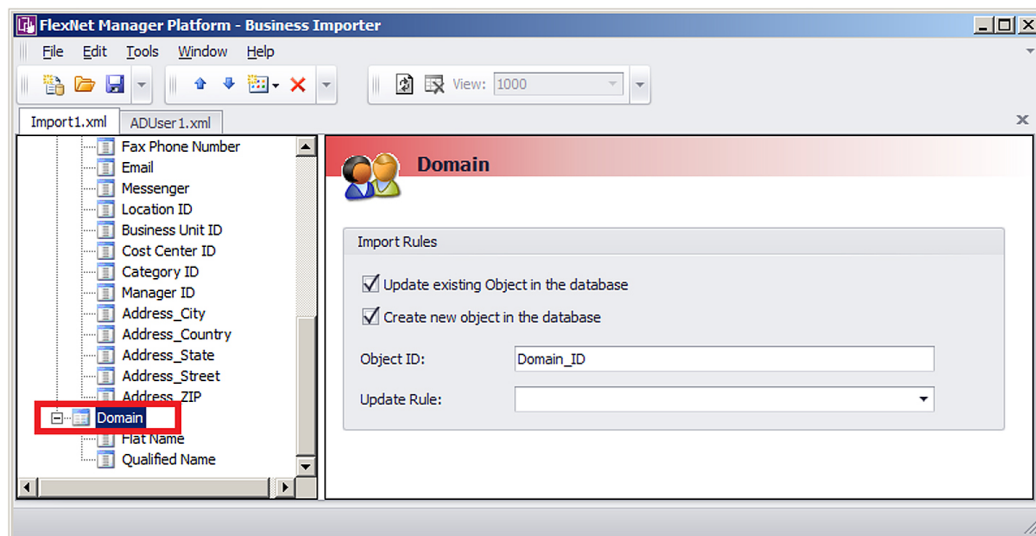
Figure 14: The import rules for an object in the compliance database

2. Complete the settings for the available fields:

Option	Description
Update existing Object in the database	The business importer always attempts to find a matching record in the compliance database for each imported record. When a matching record is found:
	• If this check box is selected (the default, and recommended), updates of the matching record may proceed according to the detailed settings you define later
	• If this check box is clear, existing compliance database objects will not be updated by this import. This setting may be useful if you wish to use this imported data to construct information that updates a different object (for example, construct information for a manager to update a user record).

Option	Description
Create new object in the database	When the business importer cannot find a matching record already in the compliance database: • If this check box is selected (the default, and recommended), a new database object will be created for the imported record • If this check box is clear, a new record will not be created for this new data in this import. You may wish to use this imported data to construct information that updates a different object (for example, construct information for a manager to update a user record).
Object ID	A unique ID for the data being imported in relation to this object. This ID appears in your finished XML file, and relates only to your import process: it is completely independent of the object's identification field in the compliance database. You may use this ID to differentiate between similar imported items, making your finished XML file easier to read and maintain. For example, if you are importing a spreadsheet of computer assets, one column might identify the leasing contract, and another identify the maintenance contract. This import requires (in addition to the asset record) the creation/update of two contract records, which you can usefully identify with this ID field (such as `LeaseContractID` and `MaintenanceContractID`).

Update Rule	For database *objects* (not relationships or queries), there are only two choices:
	
	• Leave this blank to have duplicate records in the source data silently ignored
	
	• Select `Reject duplicate records` to have duplicate records recorded in the log file.
	
	If, instead, you have selected a *relationship* between database objects, the available values depend on which relationship is selected.
	
	• **Link Contract - Asset**
	
	  • `Add new links, leave the existing ones untouched.` — (default) Never delete any existing relationships
	
	  • `Detach unfound assets from the considered contracts` — Removes asset links from contracts where assets were not found in the incoming data
	
	  • `Detach unfound contracts from considered assets` — Removes contract links from assets where contracts were not found in the incoming data
	
	  • `Detach all unfound links between the considered assets and contracts` — Removes all links not provided in the incoming data.
	
	• **Link Contract - License**
	
	  • `Add new links, leave the existing ones untouched.` — (default) Never delete any existing relationships

Option	Description

- `Detach unfound licenses from the considered contracts` — Removes license links from contracts where licenses were not found in the incoming data

- `Detach unfound contracts from the considered licenses` — Removes contract links from licenses where contracts were not found in the incoming data

- `Detach all unfound links between the considered licenses and contracts` — Removes all links not provided in the incoming data.

- **Link Purchase Order Line - License**

  - `Add new links, leave the existing ones untouched.` — (default) Never delete any existing relationships

  - `Detach all unfound licenses from the considered PO lines` — Removes license links from purchase order lines, where licenses were not found in the incoming data

  - `Detach all unfound PO lines from the considered licenses` — Removes purchase order line links from licenses, where purchase order lines were not found in the incoming data

  - `Detach all unfound links between the considered licenses and PO lines` — Removes all links not provided in the incoming data.

- **Link Purchase Order Line - Asset**

  - `Add new links, leave the existing ones untouched.` — (default) Never delete any existing relationships

  - `Detach all unfound assets from the considered PO lines` — Removes asset links from purchase order lines, where assets were not found in the incoming data

  - `Detach all unfound PO lines from the considered assets` — Removes purchase order line links from assets where purchase orders were not found in the incoming data

  - `Detach all unfound links between the considered assets and PO lines` — Removes all links not provided in the incoming data.

- **Link Payment Schedule - Asset**

  - `Add new links, leave the existing ones untouched.` — (default) Never delete any existing relationships

  - `Detach unfound assets from the considered payment schedules` — Removes asset links from payment schedules where assets were not found in the incoming data

Option	Description

- `Detach unfound payment schedules from considered assets` — Removes payment schedule links from assets where payment schedules were not found in the incoming data

- `Detach all unfound links between the considered assets and payment schedules` — Removes all links not provided in the incoming data.

- **Link Payment Schedule - License**

  - `Add new links, leave the existing ones untouched.` — (default) Never delete any existing relationships

  - `Detach unfound licenses from the considered payment schedules` — Removes license links from payment schedules where licenses were not found in the incoming data

  - `Detach unfound payment schedules from considered licenses` — Removes payment schedule links from licenses where payment schedules were not found in the incoming data

  - `Detach all unfound links between the considered licenses and payment schedules` — Removes all links not provided in the incoming data.

- **Link Users - Contracts**

  - `Add new links, leave the existing ones untouched.` — (default) Never delete any existing relationships

  - `Detach unfound users from considered contracts` — Removes user links from contracts where users were not found in the incoming data

  - `Detach unfound contracts from the considered users` — Removes contract links from users, where contracts were not found in the incoming data

  - `Detach all unfound links between the considered contracts and users` — Removes all links not provided in the incoming data.

- **Software Allocation**

  - `Add new links, leave the existing ones untouched.` — (default) Never delete any existing relationships

  - `Detach unfound software allocations from the considered computers` — Removes license allocations links from computers, where licenses were not found in the incoming data

  - `Detach unfound computers from considered software allocations` — Removes computer links from license allocations, where computers were not found in the incoming data

Option	Description
	• `Detach all unfound links between the considered computes and software allocations` — Removes all links not provided in the incoming data.
	• `Reject duplicate records` — Duplicate records are recorded in the log file, rather than being silently ignored.

Defining Import Rules for Attributes/Properties

When you have set the import rules for an item in the compliance database/>, you may set fine-grain import rules for handling imports to the individual properties of each of those objects.

1. Expand the database object in the structure tree, and select the desired property of your chosen item.

   The main page shows the available settings for this import rule. Settings are divided into groups.
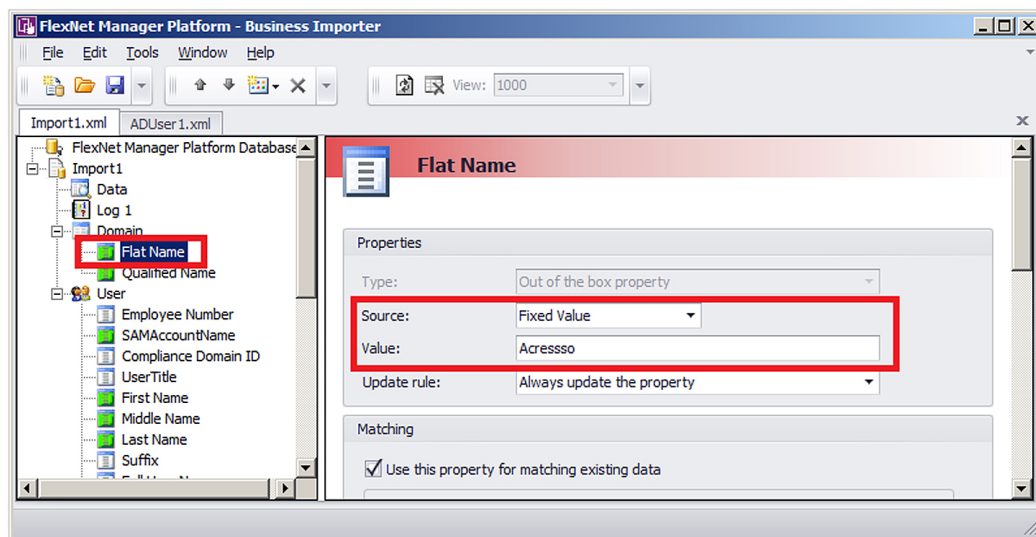
Figure 15: Import rule for a property of compliance database/> object

2. Complete the settings for the fields available in the **Properties** section of the page:

Option	Description
Type	A read-only display of the type of the property in the compliance database. Possible values are:
	• `Custom property` — A custom property defined and displayed in the UI for FlexNet Manager Suite
	• `Out of the box property` — A factory-supplied property.

Option	Description
Source	Taken together with the **Value** field, these define the source for the data to insert in this database property. In this name-value pair, this **Source** field specifies how the **Value** is to be interpreted. Available values are:

- `Field Value` — (default) The **Value** contains one of:
  - A column name in the source data
  - An ID from an item higher in the structure tree for this adapter (the output value of this item will be inserted in the database for this property).
- `Fixed Value` — A value that is specified in the **Value** field.
- `SQL Value` — The result of an SQL query. The **Value** field must contain a SQL fragment which will evaluate to a value. The SQL fragment may include columns from the data source.

Option	Description
Value	The value associated with the preceding **Source** field. Content depends on the setting for **Source** (see details above).
Update rule	Specifies what impact newly imported data is to have on information already in the compliance database/>. Possible values are:

- `Always update the property` — any incoming value (including blank) replaces any existing value
- `Never update the property` — any *existing* value (including blank) is preserved, regardless of any incoming value
- `Update only if the value is empty` — if there is no existing value, the imported value is inserted; but the imported value is ignored if there is any existing value already in the database for this property
- `Never replace an existing value with blank` — if there is a (non-blank) value in the incoming data, it replaces any existing value; but if the incoming data stream has a blank for this property, any previously existing value in the database is preserved.

3. If this property in the imported data forms part of the database key used to match existing records, select **Use this property for matching existing data**. Some database records have multi-part keys. Clear this check box when the data element does not form part of the database key in the compliance database, but is simple data. When this check box is enabled (the default), the following fields can be set.

Option	Description
If null value is found	Select one of the following values from the option list:

- `Discard the record` — The entire record is discarded when this property is empty.
- `Do not use this property for searching` — When this property is empty, it

Option	Description
	is ignored in matching for database keys. This requires that you have defined multiple properties for matching. If not (and this is the only key-matching property), this record will not match any existing data. • `Search for null value` — The property is used for searching, and matches records with a null or empty value.
Property pattern	In connected mode, this setting works in conjunction with **Matching mode** and **Value pattern**. It is relevant only when **Matching mode** is `Like` (and is otherwise ignored). For `Like` matches on key field data, this setting is a pattern to be matched in the *existing target compliance database/>*. The rules for specifying a pattern match are: • The wild card character is the percent sign (`%`). • A literal value must be enclosed in square brackets, as `[`*`value`*`]`. • The way of combining the wild card and literal value depends on the setting for **Source** (in the **Properties** group, above). For example, to create a `Like` match that expressed the concept of 'contains *value*', write the pattern in the following ways: • If **Source** is `Fixed Value`, write: `%[`*`value`*`]%` • If **Source** is `SQL Value` or `Field Value`, write: `'%' + [value] + '%'` In disconnected mode, on your inventory beacon, this setting is disabled, and is always interpreted as `[`*`value`*`]%`.
Matching mode	The type of matching to perform. This setting works in conjunction with **Property pattern** and **Value pattern**. Possible values are: • `Equal` — The value of the existing compliance database/> property must exactly match the incoming value in the imported data • `Like` — Enables matching with the use of wild cards on either side of the test, with the compliance database/> side expressed in

Option	Description
	Property pattern and the incoming data side expressed in **Value pattern**.
Value pattern	This setting works in conjunction with **Matching mode** and **Property pattern**. It is relevant only when **Matching mode** is `Like` (and is otherwise ignored). For `Like` matches on key field data, this setting is a pattern to be matched in the *data imported from the external source*. The rules for expressing the pattern to match depend on the setting for **Source**. See the details for **Property pattern**, above.

4. Where the imported data from the external source needs transformation before being inserted into the compliance database/>, complete the settings in the **Data Transformation** section of the page for each modified property. The following settings are available, and are processed in the order shown in the user interface: that is, data may be extracted using a regular expression, and the result then subject to search and replace, and so on.

Option	Description
Regular expression	Specify an expression that may be used to extract a subset of the value from the external source data. For example, to extract a flat domain name from an Active Directory record, you could write: `(?<=OU=).*? (?=,)` See also **Options** below.
Find Replace by	These two settings specify a range of substitutions that are possible per incoming property. Use these guidelines: • Separate multiple values in both fields with your choice of comma (`,`) or hash / pound signs (`#`). Use the same separator consistently for all values per property. • Spaces are significant, and are included in the processing. • Include the same number of elements to find and as replacements. Any excess in either field is ignored. For example: • **Find**: `Microsoft Corp.,Microsoft Corporation,Adobe Inc.`

Option	Description
	• **Replace by**: `Microsoft,Microsoft,Adobe`
	• *Results are: Incoming external data Value written to database* `Microsoft Corp. Microsoft Microsoft Corporation Microsoft Adobe Inc. Adobe Adobe Systems Incorporated Adobe Systems Incorporated`
Split values on	May be used only for the `groupcn` property of an enterprise group. When the incoming data expresses the group membership as a complete path, you can specify a separator character on which the string will be split into separate values. For example, if the full location path is provided as a single property containing: `USA/Boston/100 North Washington` this can be split on the slash character to form the following structure in the compliance database/>:
	🔅
	Tip • *If new records are created in the compliance database/> as a result of this import, the required parent-child links between these split elements are inserted automatically.*
	To determine the ordering of the split fields, see **Read Order** below.
Read Order	Works with **Split value on** to determine the ordering of the split values extracted from a string identifying an enterprise group. The possible values are:
	• `Forward` — (default) The left-most element is taken as the parent, with generations of children proceeding left-to-right
	• `Reverse` — The left-most element is taken as the lowest-level leaf node; the generations of children proceed from right-to-left.
Options	Specifies options for the regular expression (at the top of the page). Possible values are:
	• `CultureInvariant` — Specifies a standard convention for determining upper- and lower-case characters used in case-insensitive matching (particularly useful for matching against

Option	Description
	system resources such as account names and passwords)
	• `ECMAScript` — Specifies ESCMA script compliant behavior is enabled for the expression
	• `IgnoreCase` — Specifies case insensitive matching
	• `IgnorePatternWhitespace` — Specifies that unescaped white space is excluded from the pattern
	• `Multi Line` — Specifies multiline mode
	• `RightToLeft` — Specifies that the search moves from right to left instead of left to right
	• `SingleLine` — Specifies single-line mode.

5. If required, check or modify the settings contained in the **Advanced Properties** section of the page. The following values are available.

Option	Description
Column Name	A read-only display of the column name for this property in the compliance database/> in FlexNet Manager Suite.
Data Type	A read-only display of the data type of this property in the compliance database/> in FlexNet Manager Suite.
Length	A read-only display of the length of this property in the compliance database/> in FlexNet Manager Suite.
If value is missing	Determines the behavior of the business importer in cases where the column for this property is entirely missing from the source data. When you are designing your own adapter, a missing column probably means that something has gone grossly wrong, and your import should fail, in order to draw your attention to the problem. The other values are more useful for some factory-supplied adapters for spreadsheet data, where there is less control over the columns in the source data. Possible values are:

Option	Description
	• `Do nothing (the import will fail)` — (default) Leave this value selected for most adapters, as a failed import will alert you to a problem in the source data
	• `Remove the property from the import` — The import will proceed, but (silently) no values will be recorded for this property
	• `Remove the object from the import` — The import will proceed, but (silently) no instances of the parent object (of which this property is an attribute) will be created or updated.
Format	Leave this option list blank if the format of dates and times in the imported data file is the same as the local setting on the application server running FlexNet Manager Suite. If the formats are different, choose an option from the list that defines the format used in the input data file.
Use the following query to match existing computers	This field only displays for assets.
	Specific types of assets in FlexNet Manager Suite (for example workstations, servers), can be attached to a computer. When the Business Importer creates these types of assets in the FlexNet Manager Suite database, it performs a lookup against computer records that are not already attached to an asset. It tries to match assets to computers with the same serial numbers. If a match is found, the computer is linked to the asset.
	Computers set to a status of `Ignored` are not included in the matching process.
	Set this field in one of the following ways:
	• Leave the field blank to use the default computer matching behavior.
	• Type a single space to disable computer matching. No computer matching will take place.
	• Enter an SQL statement to customize the computer matching behaviour.
	When entering an SQL statement, the following keywords can be used:

Option	Description
	• `[TemporaryTableName]` - The name of the temporary or physical table used by the import.
	• `[OutputFIeld]` - The name of the field containing the Asset ID values for existing records or new records created.
	• `[ImportID]` - The ID of the record in the ECMImportLog_Import table processing the import
	• `[ImportObjectID]` - The ID of the record in the ECMImportLog_Object table processing the computer object.
	If you build specific logic to perform the computer matching, the list of newly created Asset IDs can be retrieved with the following query:
	`"Select [OutputField] from [TemporaryTableName] where created =1"`
	The SQL procedure can also return details of the number of computers affected. This number is logged in the ECMImportLOG_Object table.

6. As you make your changes on this page, your specification is saved in memory. When you are ready, click the Save button in the tool bar, or choose one of the saving options from the **File** menu.

Creating Custom SQL

Because you are responsible for your own database, you have the freedom to add custom SQL that modifies your compliance database during the import of business data.

At any stage in your adapter, which runs from top to bottom down the structure tree, you can insert some custom SQL to modify (or make log entries for) the data that has (so far) been written into the compliance database in FlexNet Manager Suite. You may include:

• SQL statements

• Stored procedures.

Any custom SQL is run as part of the same transaction as the import of the external data.

1. Right-click your adapter's name in the structure tree.

2. In the context menu, select **Add New Item**, and from the submenu select **Custom Query**.

3. To give your query a meaningful name (making it easier to read and maintain in the finished adapter XML file):

    a) Right click the new **Custom Query** entry in the structure tree

    b) Select **Rename** from the context menu

c) Provide a new name.

4. Enter your SQL statement into the **Query Text** field.

💡

Tip • You may include the keyword `[LOG_IMPORT_ID]` *to return the current value of the* `ImportID` *column in the* `ECMImportLog_Summary` *table.*

For example:

- To run a stored procedure called *MyStoredProcedure* (optionally with parameters), enter

```
EXEC MyStoredProcedure Parameter1,...
```

- To post an entry in the log for a custom object (which will not otherwise be logged), enter

```
Insert into ECMImportLog_Object([ImportID], [ObjectName], [StartDate],
                                [EndDate], [ObjectType], [Status]) values
 ([LOG_IMPORT_ID],
                                'My Custom Object', getdate(),
                                getdate(),'Custom', 1)
```

5. In the **Timeout** field, set the number of seconds that the business importer should wait before giving up waiting for a response to the custom SQL statement. The following values have special meaning:

- `-1` means use the default time out for SQL Server

- `0` means there is no limit and the business importer will wait indefinitely for the query to finish.

6. If you wish to write a custom message in the log file as part of this custom SQL, select the **Send the internal log ID as a parameter** check box. When checked, the `@FinImporterLogID` parameter value is automatically injected to the stored procedure as it is executed. The value of this parameter is the current record ID from the `FinImporterLog` table (automatically created).

Adding Custom Properties

Business adapters that run in connected mode (that is, on your central application server) can create custom properties in your compliance database on the fly.

FlexNet Manager Suite allows the addition of custom properties to existing objects in the compliance database. You can do this using the web interface. However, since it is very likely that the need for custom properties becomes apparent when integrating external data sources, you can also specify custom properties as a part of building a business adapter, as long as this adapter runs in connected mode on your operations server. The first time that the business importer runs this business adapter, the new custom property is created. On subsequent imports, data can be added to the custom property in the same way as for all other properties (the adapter does not have to be modified to remove the creation of the custom property).

💡

Tip • Remember that you must create the custom property before you can populate it with data — meaning that the database object with the custom property attached must be high in the structure tree.

Note • *Any business adapter that creates custom properties can be run only on the application server. Once custom properties are created (through either means), you can also populate those properties with values through another business adapter running in disconnected mode on the inventory beacon.*

1. In the structure tree on the left of the Business Adapter Studio, right-click the compliance database object for which you want to create a custom property.

2. From the context menu, select **Add New Item**.

3. From the submenu, select **Custom Property**.

4. To give your custom property the name that will be created in the FlexNet Manager Suite compliance database:

    a) Right-click the new entry **Custom Property *n*** in the structure view.

    b) Select **Rename**.

    c) Type the new name, and press `Return`.

    The details in the main data page are the same for this custom property as for all other properties (see *Defining Import Rules for Attributes/Properties* on page 285).

Triggering Immediate Update of the BAS Data Model

The data model exposed to the Business Adapter Studio installed on your inventory beacon(s) is updated by a scheduled task (`Regenerate Business Import config`) that runs overnight (by default, at 4am central server time). Therefore, if you add custom properties to FlexNet Manager Suite, you normally need to wait until next day before you can create a business adapter that loads data into your new custom field.

Alternatively, you can use the following process to trigger an immediate update to the data model for the Business Adapter Studio. This allows you to continue development from custom properties straight on to a custom business adapter that populates those properties.

1. On the batch server, open a Command Window.

2. Navigate to:

    ```
    InstallDir\DotNet\bin\
    ```

    The default value is

    ```
    C:\Program Files (x86)\Flexera Software\FlexNet Manager Platform\DotNet\bin\
    ```

3. Execute the following command:

    ```
    BatchProcessTaskConsole.exe run BusinessAdapterConfig
    ```

    This launches the task that generates the updated data model for the Business Adapter Studio. To check when it is finished, run:

    ```
    BatchProcessTaskConsole.exe list-tasks
    ```

While the task is running, `BusinessAdapterConfig` is visible in the task list, and it disappears within a few minutes, when the task is successful. The updated data model is then automatically collected by all inventory beacons when they "phone home" for updates. By default, this happens every 15 minutes, but the interval is configured in the web interface under **Discovery & Inventory** >**Settings**, under the **Beacon settings** section. Thereafter, restarting the Business Adapter Studio forces it to reload the data model.

4. After the propagation time, restart the Business Adapter Studio on your chosen inventory beacon. If you are running the Business Adapter Studio in connected mode on your central server, simply exit and restart. If you are running the Business Adapter Studio in disconnected mode on an inventory beacon, use this process::

   a) If the Business Adapter Studio is already open on your inventory beacon, you must exit and restart the entire FlexNet Beacon interface.

   b) In the FlexNet Beacon interface, navigate to the **Business Importer** page.

   c) Edit an existing import, or create a new one, and click **Edit adapter...**.
   Your newly-created custom properties are included in the list of available properties with a distinctive icon.

Testing and Diagnosis for Your Business Adapter

In connected mode, before running any import (even to a test database, let alone a production environment), you can run a simulation that will show the projected results of the business importer running your new adaptor.

In connected mode, you will also want to create log files for your imports, and possibly run tracing. All these topics are included in this section.

Testing and diagnosis options are limited for business adapters running in disconnected mode on your inventory beacon. After running the adapter against the source database, you can inspect the archive package that will be uploaded to the central application server. This package is saved in `Program Data\Flexera Software \Beacon\IntermediateData\`, and is a zip archive containing the following:

• A file DDI.xml that represents the business adapter (without its connection strings), so that you can see the steps than run in your adapter

• A manifest that includes a result code from running your adapter, and any error messages

• An XML file of the collected data.

On the inventory beacon, you may also examine the log file for the beacon engine, which includes results of uploading the intermediate package. This is saved in `%ProgramData%\Flexera Software\Compliance \BeaconEngine`. You may search in the log file for the name of your business adapter to find steps relating to it.

Note • *There is no specific presentation in the compliance console (the web UI) of the results of importing your business adapter data. Look for collected information displayed in appropriate lists after the import is completed.*

Of course, the impact on your compliance position that results from this imported business information is shown only after the next inventory import and reconciliation.

Specifying a Log File

Log files help you monitor the operations of your business adapter.

You can specify a log file that is used for all imports using this business adapter. The log that you specify is updated automatically by the business importer, overwriting the previous log file on each occasion.

1. Right-click on the adapter name in the structure tree.

2. From the context menu, select **Add New Item**, and from the submenu select **Log**.

   A page of log file settings appears, populated with default values.
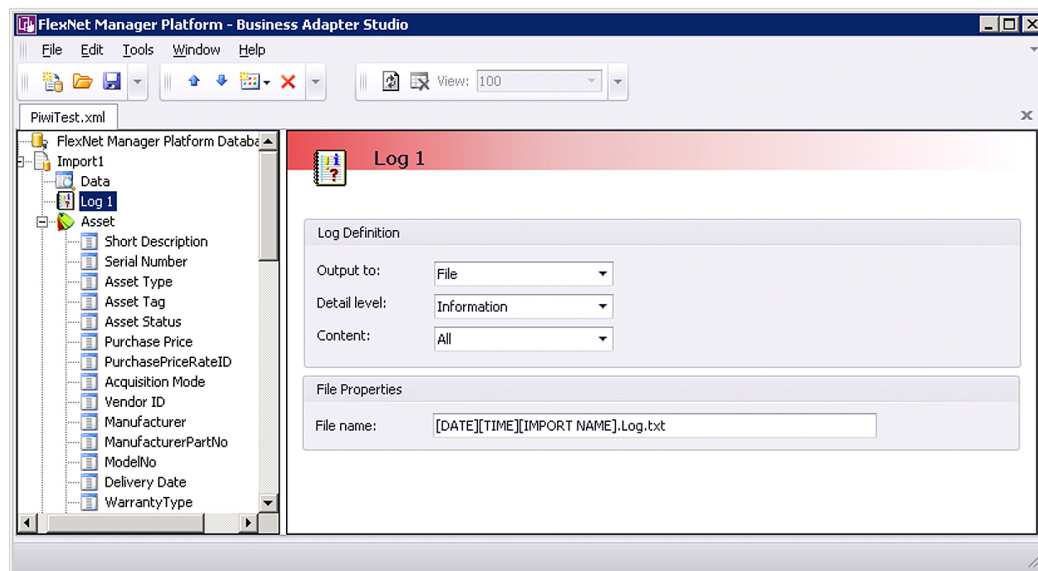
Figure 16: The default values for a log file

3. Adjust the following **Log Definition** settings to your requirements.

Setting	Notes
Output to	The type of log output to generate. Possible values are: • `Console` — Output appears on the console of the test computer during development of the business adapter. It is likely to be less useful when the production version of the business adapter is running on the inventory beacon or the application server. • `File` — Output will be saved to a text-based log file. When you select this option, additional fields appear at the bottom of the page to specify the file details. • `Database` — In connected mode only (not available on the inventory beacon), the log is written into the compliance database, in the following tables (available

Setting	Notes
	only in Microsoft SQL Server Management Studio, and not in the user interface for FlexNet Manager Suite): • `ECMImportLog_Detail` — A record is written here for every database record that has tracing turned on (see *Tracing* on page 298) when the record is rejected, created, updated or deleted. • `ECMImportLog_Object` — A record is written here for each compliance database standard object that is created or updated during an import. Custom objects are not logged automatically, although you can cause a log entry to be added for a custom object (see *Creating Custom SQL* on page 292). • `ECMImportLog_Summary` — A record is written here each time the business importer is started, either for a data import or for a simulation during testing. The record includes a `Status` result of `0` for a task not completed (error), and `1` for success. • `Tcp/IP socket` — Output is written to a TCP/IP socket. When you select this option, additional fields appear at the bottom of the page to specify the TCP/IP details.
Detail level	The depth of logging to provide. Choosing a given level of logging includes information for all higher levels in this ordered list: • `Silent` — (highest value) No details are recorded for individual database objects, although a record of the overall import/simulation still appears in the summary table (for database logging). • `Critical` • `Errors` • `Warnings` • `Information` • `Debug` — (lowest value) The most detailed logging information is written.
Content	Master control for what is written to the log. The settings are: • `Detail` — Writes information for each object to the log according to the level specified in the **Detail level** setting (above) • `Summary` — At the end of the import, writes a summary for each affected object. • `All` — Combination of the `Summary` and `Detail` levels.

4. If you selected **Output to** `File`, specify the name of the log file in the **File name** setting.

   • If you clear this field, the default file name is `MBI.log.txt`.

   • You may include UNC path definitions, or relative paths starting from the directory where the business importer executable is running.

- When no file path is specified, the default is a subfolder `Log`, below the `business importer` folder.

Remember that a log file with the same name is overwritten at each import. You may wish to use the following placeholder variables to create a unique file name for each pass at least during test and development (log files with unique names are not automatically cleaned up, and you assume that responsibility):

- `[DATE]` — the date the import was started (formatted as *yyyymmdd*)

- `[TIME]` — the time the import was started (formatted as *hhmmss*)

- `[IMPORT NAME]` — the name of the adapter used for this import, taken from the name set for the adapter in the structure tree of the Business Adapter Studio.

5. If you selected **Output to** `Tcp/IP Socket`, specify the settings in the **TCP/IP Socket Properties** section that appears:

- **Server** — The server name where the listener is installed. For the same computer on which the business importer is running, use `localhost`.

- **Port number** — The port where the listener is waiting.

Tracing

Tracing is available only for those business adapters running in connected mode (that is, on your central application server).

Tracing adds another stream of information, additional to logging outputs, to make debugging easier. It is controlled from the adapter node in the structure tree. Tracing outputs are written to the `ECMImportLog_Detail` table in the compliance database under FlexNet Manager Suite, regardless of the logging output type that you selected. You can inspect the rows of this table in Microsoft SQL Server Management Studio.

1. Select the adaptor node in the structure tree, and scroll down the settings page to see the **Tracing** group.
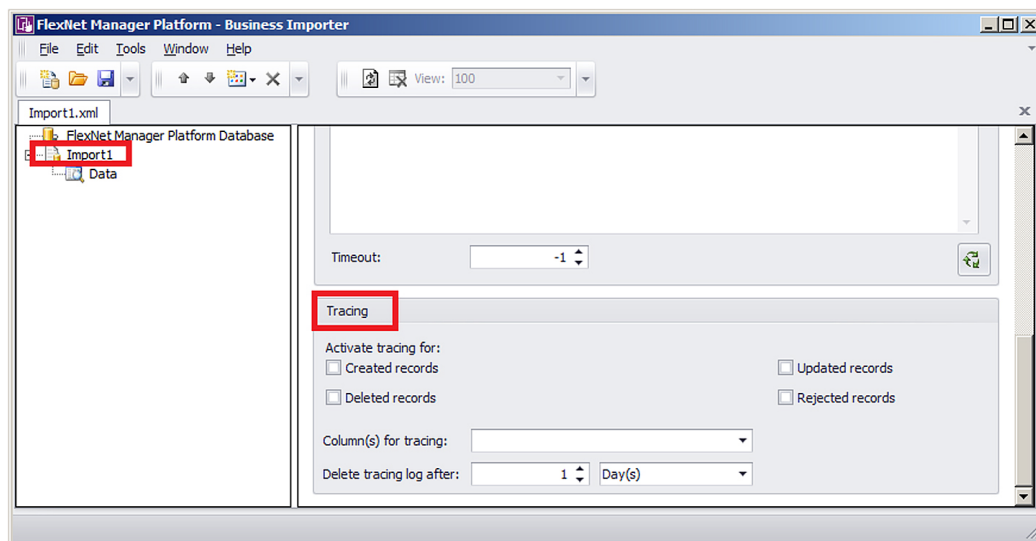
Figure 17: Tracing settings are at the bottom of the settings page for the adapter

2. Adjust the following settings for tracing:

Setting	Notes
Activate tracing for	Select as many of the following check boxes as needed for the tracing you require for debugging your adapter. Note that if no check boxes in this group are selected, no tracing records are created. You may create a tracing entry each time that any record in the compliance database is created, updated, or deleted; or when a record in the source data is rejected. **Rejected records** are incoming data assessed for import but not accepted. A common reason for rejection is that a column (such as a serial number) has been specified for database matching, but that column is blank for a particular record.
Column(s) for tracing	By default, tracing identifies affected records only by their count number within the flow of incoming data from the external source. As this is not a convenient aid to debugging, you may also pick one or more columns from the incoming data to help identify the tracing records. For example, if you selected a column called `Serial No` from this option list of the available columns in the data source, and the computer with serial number 19283645 was rejected, the tracing would identify `Record 238, Serial No 19283645` This is more convenient for identifying the offending record. Best practice is to use an identifying column that contains unique values and is always populated. You may either: • Select a single column name (in the external data source) from this option list • Directly enter multiple column names in the field, enclosing each in square brackets and separating the bracketed values with a comma.

Simulating an Import of a Business Adapter

When you run a simulation, Business Adapter Studio processes all of the data as if an import were taking place, and lists the number of records that would have been processed and the number of records that would be rejected.

This means that you can review the results of the simulated import, and, if necessary, refine the connection settings before doing an actual import and changing the FlexNet Manager Suite data.

A simulation does not load data into the object records in the compliance database in FlexNet Manager Suite, but it does update the history tables for every object that would have been affected were the import actually performed, to record that a simulation has taken place. These history records are visible in the **History** tabs of each object's property sheet.

Once you are satisfied with the results of a simulated import, you can perform a real import. Even after performing a simulation, it is best practice to run the first import on a test database rather than in your production environment.

1. Ensure you have selected the tab for the adapter you wish to test.

2. Select the **Tools** > **Simulate** menu options.

The **Simulate** page is displayed. Do not change the default parameters.
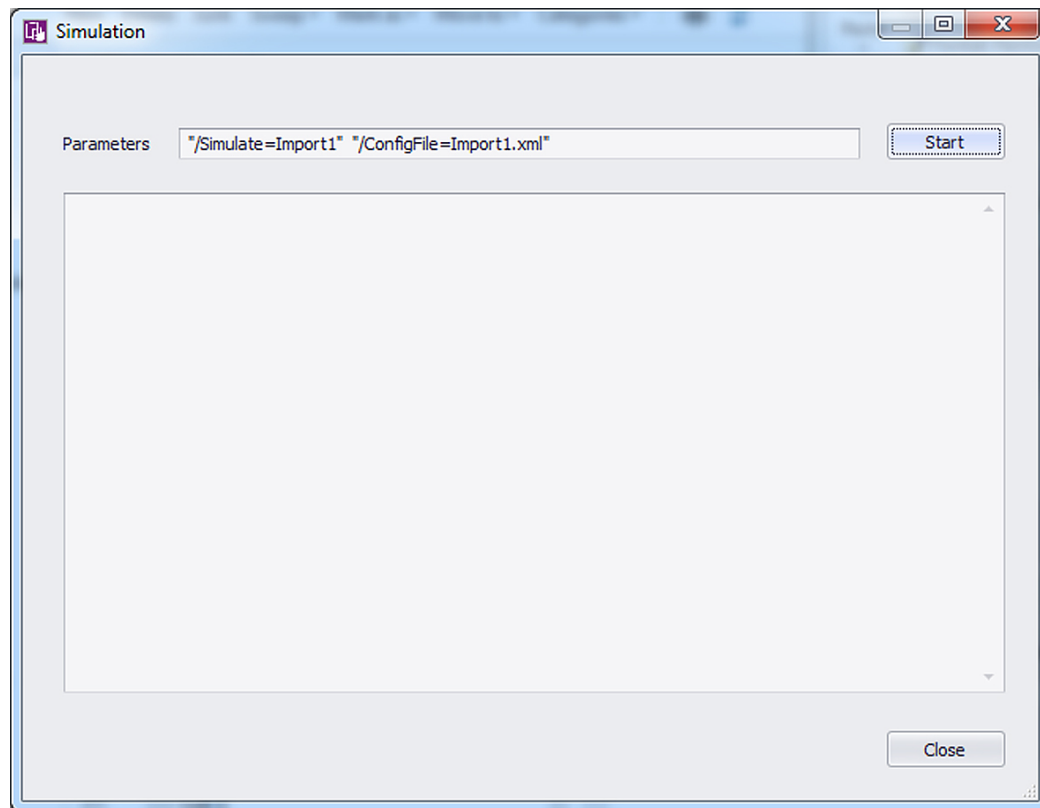

Figure 18: Run a simulation to check that the import file can be processed without affecting the FlexNet Manager Suite data

3. Click **Start**.

The simulated import is processed, and progress messages display on the window. When the import is complete, a summary of the records that could be processed in the simulation is displayed.
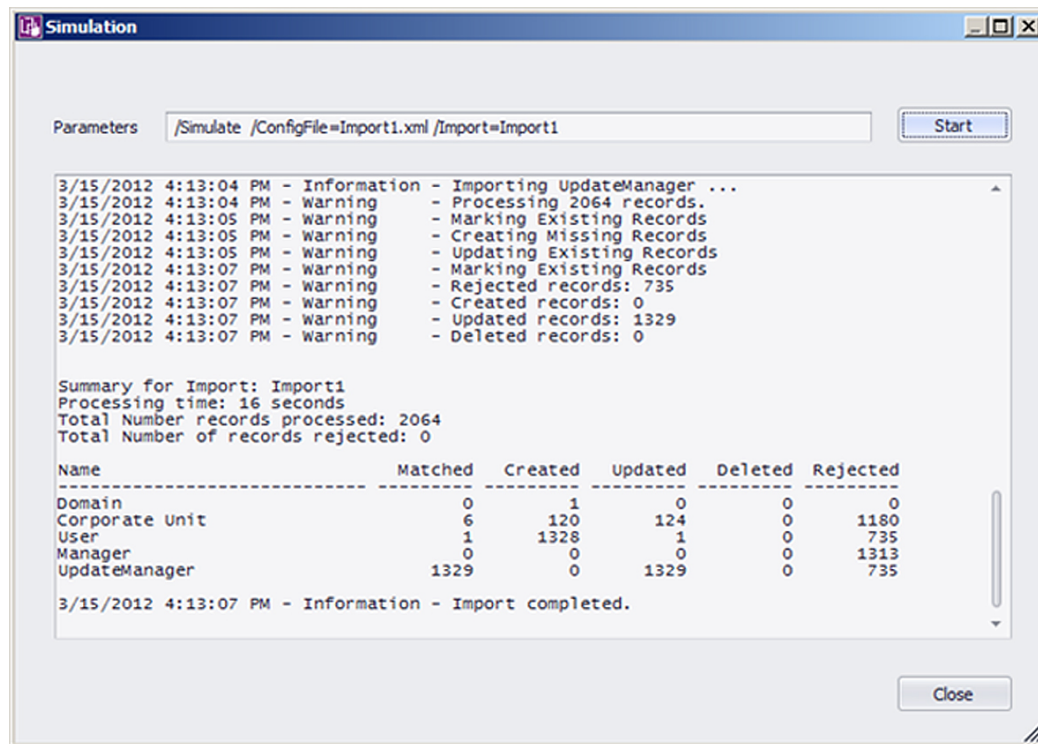
Figure 19: Review the results of the simulated process

4. Review the results. Check for any records rejected, and whether the correct number of records were processed as expected.

💡

Tip • *The most common reason for an incoming record to be rejected is that it lacks a value in a field that has been nominated for matching with existing records in the compliance database.*

5. When you have finished reviewing the results, click **Close**.

Comparing Result of Simulated Imports

Once you have run several simulations, you may want to compare the overall results.

A history of simulations is available, combined with the history of data imports.

1. With the tab selected for the correct adapter, select the **Tools** > **View History** menu options.

The **History for ...** page is displayed.

Figure 20: A list of past imports and simulations is displayed

2. To view details of the records imported (or simulated) during one of the actions listed on the page, expand the icon to the left of the import action.

   Details of each record imported or updated (or the simulation of that action) are displayed.

3. When you have finished reviewing the history, click **Close** to close the window.

4. If the results are as expected, your connection is ready for you to perform a real import. If the results are not as expected, you may need to modify the configuration for this adapter. See *Linking Data Imports to FlexNet Manager Suite* on page 277 for more guidance.

Troubleshooting Business Adapters

Here are some possible issues and causes. Please advise any other cases that should appear here in future.

Issue	Notes
Imported data does not appear in custom views	There may be dependencies between different data fields. For example, if you are importing details about purchase orders than include prices, each price must have a corresponding rate (currency) identified. Without this correspondence, the numeric values are blanked out of the custom view. (For example, in the product schema, see `UnitPrice` and `UnitPriceRateID` in the `PurchaseOrderDetail` table.)

Issue	Notes
Monetary values appear without any currency units	The rate ID for this value is missing from your imported data.

Importing Data With Your Business Adapter

You have tested and debugged your business adapter. You have run a simulation and checked the imported values are as expected. Now you are ready to import data into your operations databases.

You may:

- Perform a single import for an adapter running in connected mode, where that import is controlled by the Business Adapter Studio

- Examine the history of imports you have run from the Business Adapter Studio, in connected mode

- Schedule regular imports for operations, which do not require the Business Adapter Studio

- Trigger a special, additional import from within the web interface, independent of the Business Adapter Studio.

Running an Import from the Business Adapter Studio

When you have simulated your import and you are satisfied with the results, you can perform a real import.

Important •

When you import data, you make live changes to the FlexNet Manager Suite database. Incorrect settings may result in important data being deleted or modified. You should always test your import in a test environment or pre-production environment before importing into the live data.

Important •

Always perform a full backup of the FlexNet Manager Suite database before performing an import. This will allow you to rollback the database to its original state if the imported data is corrupted or configuration of the import data is incorrect.

1. Make sure you have simulated the import and have confirmed that the import will perform the required tasks.
2. Back up the FlexNet Manager Suite database.
3. With the tab for the correct adapter selected, select the **Tools** > **Import** menu options.

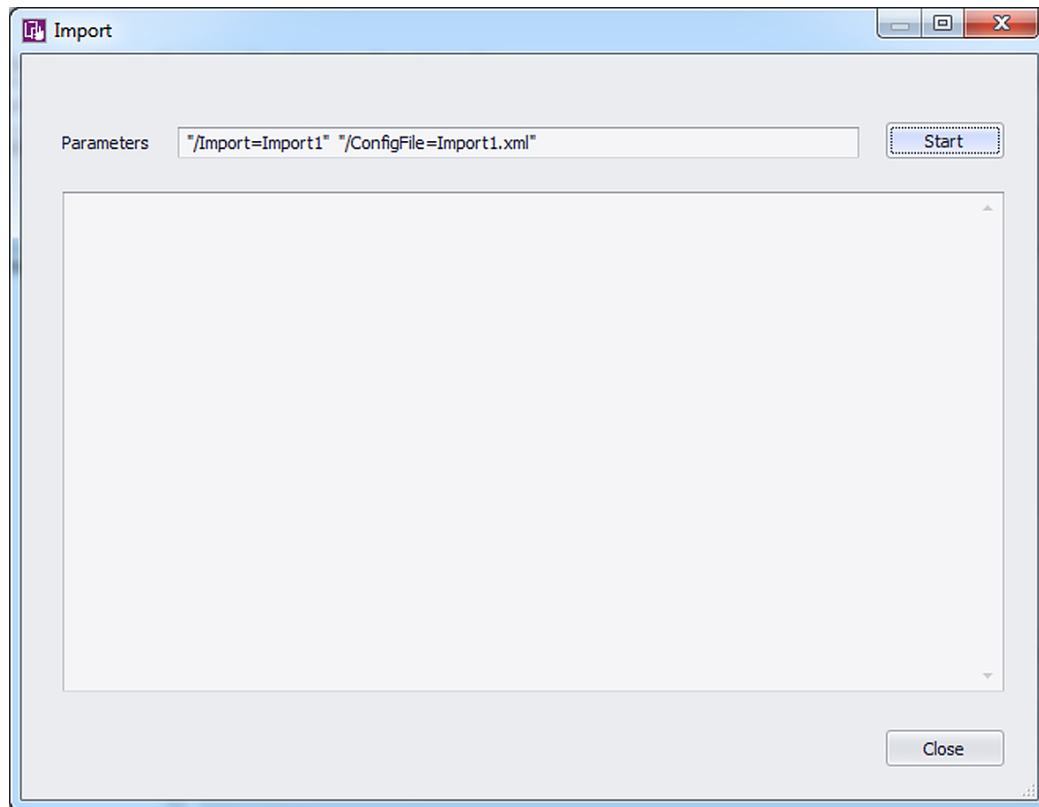   The **Import** page is displayed.

Figure 21: Check the import parameters and then start the import

4. Click **Start**.
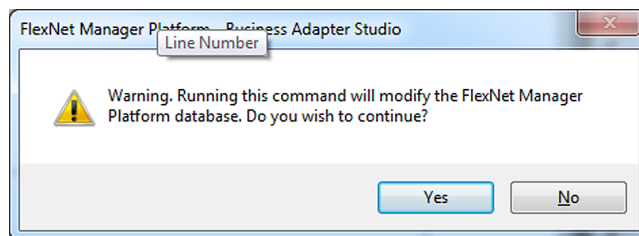
A warning message is displayed.

Figure 22: Confirm that you want to perform an import

5. Click **Yes** to confirm that you want to perform an import. (Alternatively, click **No** if you want to cancel the import.)

The import is processed, and progress messages display on the window. When the import is complete, a summary of the records processed is displayed.
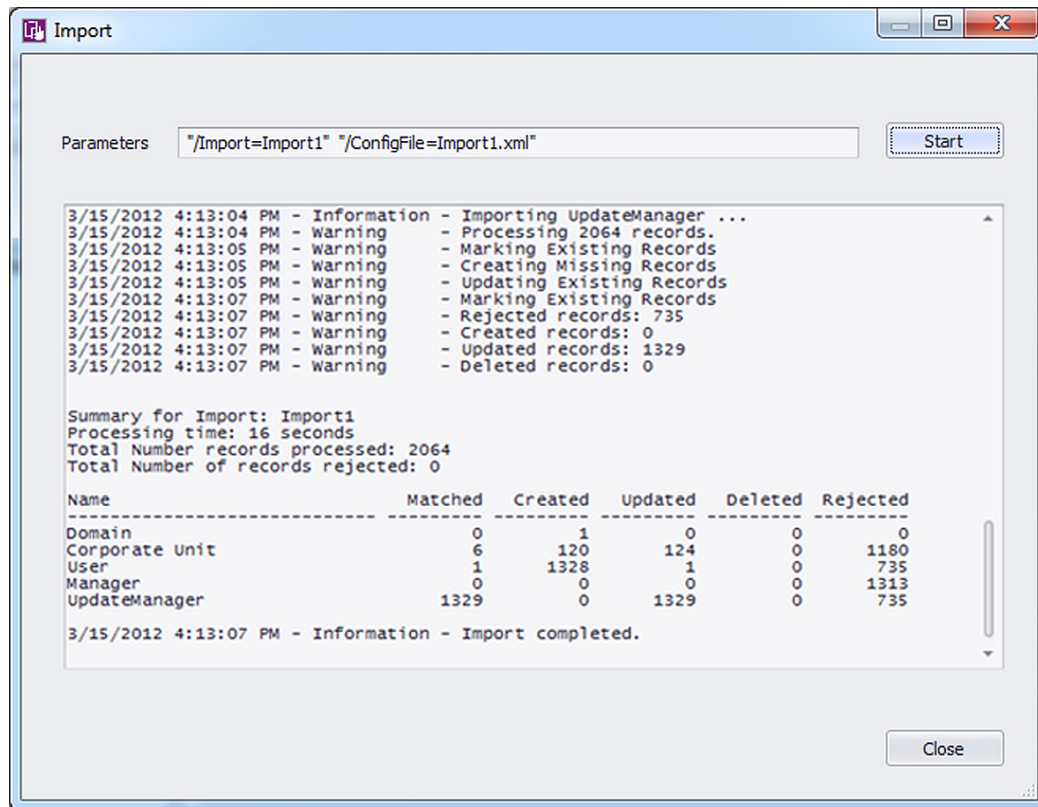
Figure 23: Review the results of the import process

6. Review the results. Check for any records rejected, and whether the correct number of records were processed as expected.

7. When you have finished reviewing the results, click **Close**.

Tip • If you expect to run this data import process more than once, it is best practise to set up a scheduled task to run the import on a regular basis. Use the Windows Scheduled Tasks system tool to create a scheduled task.

To Review Past Imports in Business Adapter Studio

You can only review the history of those imports triggered in Business Adapter Studio. For regular imports, consider logging.

If you have run multiple imports of business information through Business Adapter Studio, you can review the history of your results.

1. With the appropriate tab displaying your adapter, select the **Tools** > **View History** menu options.

   The **History for *adapterName*** page is displayed.

Figure 24: A list of past imports and simulation imports is displayed

2. To view details of the records imported (or simulated) during one of the actions listed on the page, expand the icon to the left of the import action.

   Details of each record imported or updated (or the simulation of that action) are displayed.

3. When you have finished reviewing the history, click **Close** to close the window.

Setting Up Regular Imports (Connected Mode)

On your central application server, create scheduled tasks to run regular business imports in connected mode.

Many external data sources require regular imports to the compliance database to pick up later additions and modifications to the source data. For example, imports from your purchasing system must be repeated on a regular schedule to pick up new purchases that modify your overall license compliance.

To repeat the import using your new business adapter on a regular basis, create a standard Windows scheduled task on the application server.

Tip • *If you need to move your business adapter file from a development environment into your production environment, remember that you may need to modify the connection strings to suit.*

The default command line for your scheduled task has the following form:

```
InstallationPath\mgsbi.exe /Import=AdapterName /ConfigFile=XMLFileName
```

where the placeholders represent:

InstallationPath

> The location where the business importer executable is installed on the compliance server. By default this is `C:\Program Files\ManagerSoft\DotNet\bin`.

AdapterName

> The name displayed for your adapter in the structure tree of the Business Adapter Studio. For information about naming your adapter, see *Renaming a Business Adapter* on page 256 (This name is also visible as the `Name` attribute for the `<Import>` element in the saved XML file.) Remember that if the adapter name includes spaces, you must enclose it in double quotation marks.

XMLFileName

> The name under which you saved your XML file (see *Saving Business Adapters* on page 257). Once again, if the file name includes spaces, enclose it in double quotation marks.
>
> ---
>
> *Tip • If you do not specify an XML file name, the default file `MGSBI.xml` is searched for the adapter.*