FLEXERA SOFTWARE

FLEXERA SOFTWARE®
FlexNet Manager® Suite

FlexNet Manager Suite System Reference

Legal Information

Document Name: FlexNet Manager Suite 2016 R1 SP1 System Reference (for cloud implementations)

Part Number: FMS-12.1.0-SR04

Product Release Date: December 19, 2016

Copyright Notice

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

Intellectual Property

Restricted Rights Legend

System Reference

This document gathers together a range of reference material for FlexNet Manager Suite release 2016 R1 SP1. 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.

Contents

1

Adding Custom Properties

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.

Note: *If you are using a cloud-based implementation of FlexNet Manager Suite, you do not have direct access to the database, and you cannot "do it yourself". However, you may use this section to understand and specify your requirements. You can then send a support request to Flexera Software specifying all the details you require, and your custom property will be added on your behalf.*

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 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 (central server time). This means waiting until next day before importing values with a business adapter to your new custom field.

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.*

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.

Note: *If you are using a cloud-based implementation of FlexNet Manager Suite, you do not have direct access to the database, and you cannot "do it yourself". However, you may use this section to understand and specify your requirements. You can then send a support request to Flexera Software specifying all the details you require, and your custom property will be added on your behalf.*

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 **5.** Chair **6.** Printer **7.** Router **8.** Switch **9.** Telephone **10.** Cellphone **11.** Laptop **12.** Mobile Device

Object	TargetTypeID	TargetSubTypeID
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 **5.** Other **6.** Software Upgrade **7.** Software Maintenance **8.** Disk Kit **9.** Hardware Maintenance

Object	TargetTypeID	TargetSubTypeID
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 **26.** IBM Authorized User **27.** IBM Concurrent User **28.** IBM Floating User **29.** Custom Metric **30.** One Point Per Processor

Object	TargetTypeID	TargetSubTypeID
		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	
Vendor	24	

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.

Note: *If you are using a cloud-based implementation of FlexNet Manager Suite, you do not have direct access to the database, and you cannot "do it yourself". However, you may use this section to understand and specify your requirements. You can then send a support request to Flexera Software specifying all the details you require, and your custom property will be added on your behalf.*

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.
Drop-down 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.

Note: *If you are using a cloud-based implementation of FlexNet Manager Suite, you do not have direct access to the database, and you cannot "do it yourself". However, you may use this section to understand and specify your requirements. You can then send a support request to Flexera Software specifying all the details you require, and your custom property will be added on your behalf.*

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.

Note: *If you are using a cloud-based implementation of FlexNet Manager Suite, you do not have direct access to the database, and you cannot "do it yourself". However, you may use this section to understand and specify your requirements. You can then send a support request to Flexera Software specifying all the details you require, and your custom property will be added on your behalf.*

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
	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.

Note: *If you are using a cloud-based implementation of FlexNet Manager Suite, you do not have direct access to the database, and you cannot "do it yourself". However, you may use this section to understand and specify your requirements. You can then send a support request to Flexera Software specifying all the details you require, and your custom property will be added on your behalf.*

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

Section	Control	Internal Name
	Manufacturer	Manufacturer
	Part number	PartNumber
	Model	AssetModel
	Master asset	MasterAssetName
	Status	Status
	Category	CategoryPath
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

Section	Control	Internal Name
	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

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

Section	Control	Internal Name
	Buyout	BuyOut
	Payment	Payment
	Period	Period
Lease Termination *(see note)*		Section_LeaseTermination
	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.

Note: *If you are using a cloud-based implementation of FlexNet Manager Suite, you do not have direct access to the database, and you cannot "do it yourself". However, you may use this section to understand and specify your requirements. You can then send a support request to Flexera Software specifying all the details you require, and your custom property will be added on your behalf.*

Tab label: General

Database identity: Tab_Summary.

Section	Control	Internal Name
General		Section_Summary
	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

Section	Control	Internal Name
	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.

Note: *If you are using a cloud-based implementation of FlexNet Manager Suite, you do not have direct access to the database, and you cannot "do it yourself". However, you may use this section to understand and specify your requirements. You can then send a support request to Flexera Software specifying all the details you require, and your custom property will be added on your behalf.*

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
	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.

Note: *If you are using a cloud-based implementation of FlexNet Manager Suite, you do not have direct access to the database, and you cannot "do it yourself". However, you may use this section to understand and specify your requirements. You can then send a support request to Flexera Software specifying all the details you require, and your custom property will be added on your behalf.*

Tab label: Compliance

Database identity: Tab_Compliance.

Section	Control	Internal Name
Compliance		Section_Compliance
	Compliance status	ComplianceStatus
	Shortfall/Availability	Available
	Breach reason	BreachReason

Section	Control	Internal Name
Entitlements and consumption		EntitlementsAndConsumption
	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.

Note: *If you are using a cloud-based implementation of FlexNet Manager Suite, you do not have direct access to the database, and you cannot "do it yourself". However, you may use this section to understand and specify your requirements. You can then send a support request to Flexera Software specifying all the details you require, and your custom property will be added on your behalf.*

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

Section	Control	Internal Name
	Status	ItemStatus
	Purchase type	Type
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.

Note: *If you are using a cloud-based implementation of FlexNet Manager Suite, you do not have direct access to the database, and you cannot "do it yourself". However, you may use this section to understand and specify your requirements. You can then send a support request to Flexera Software specifying all the details you require, and your custom property will be added on your behalf.*

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

Section	Control	Internal Name
	Phone	Phone
	Fax	Fax
	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

Section	Control	Internal Name
Contracts		Section_Contracts
	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.

Note: *If you are using a cloud-based implementation of FlexNet Manager Suite, you do not have direct access to the database, and you cannot "do it yourself". However, you may use this section to understand and specify your requirements. You can then send a support request to Flexera Software specifying all the details you require, and your custom property will be added on your behalf.*

Tab label: General

Database identity: Tab_General.

Section	Control	Internal Name
Identification		Section_Identification
	Name	Name
Contact information		Section_ContactInformation
	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.

Note: *If you are using a cloud-based implementation of FlexNet Manager Suite, you do not have direct access to the database, and you cannot "do it yourself". However, you may use this section to understand and specify your requirements. You can then send a support request to Flexera Software specifying all the details you require, and your custom property will be added on your behalf.*

Syntax:

```
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.

@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.
@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.
@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. 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.

Creating a New Section Within a Tab

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

Note: *If you are using a cloud-based implementation of FlexNet Manager Suite, you do not have direct access to the database, and you cannot "do it yourself". However, you may use this section to understand and specify your requirements. You can then send a support request to Flexera Software specifying all the details you require, and your custom property will be added on your behalf.*

Syntax:

```
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.
@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.

@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.
@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.
@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. 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. 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.

Note: *If you are using a cloud-based implementation of FlexNet Manager Suite, you do not have direct access to the database, and you cannot "do it yourself". However, you may use this section to understand and specify your requirements. You can then send a support request to Flexera Software specifying all the details you require, and your custom property will be added on your behalf.*

Syntax:

```
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.

@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.
@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.
@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.
@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. For details about creating custom tabs, see Creating a New Properties Tab.

@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. 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.
@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.
@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.
@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.
@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 drop-down list in the web interface. Example: `@DataSource = ', Apples, Oranges, Ripe Pears, Tangerines'` This example creates a drop-down 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 drop-down list (when `UIFieldTypeID = 8`), it is not possible to localize the values for the individual options within the custom drop-down 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 options, including added custom options; whereas drop-down lists that are in entirety custom cannot be localized.)*

@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.

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.

Note: *If you are using a cloud-based implementation of FlexNet Manager Suite, you do not have direct access to the database, and you cannot "do it yourself". However, you may use this section to understand and specify your requirements. You can then send a support request to Flexera Software specifying all the details you require, and your custom property will be added on your behalf.*

Syntax:

```
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.

Note: *If you are using a cloud-based implementation of FlexNet Manager Suite, you do not have direct access to the database, and you cannot "do it yourself". However, you may use this section to understand and specify*

your requirements. You can then send a support request to Flexera Software specifying all the details you require, and your custom property will be added on your behalf.

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.LicenseTab.Charges`, containing `My.LicenseSection.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.

Syntax:

```
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.
@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.

2

Importing Inventory Spreadsheets and CSV Files

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
- CAL usage inventory, called access evidence (see the *FlexNet Manager Suite System Reference Guide* for details)
- 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.

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.)

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. 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** drop-down 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** drop-down 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). 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). 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*** *drop-down 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.*

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

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

   c. 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

Introduction to Client Access License

A Client Access License (CAL) is a software license that is required to access the services of certain server products like `Microsoft Exchange Server` or `Microsoft SharePoint Server`. A CAL is required for each accessing user or device that accesses the server product unless a processor-based or core-based license is procured for the accessed server application, or an External Connector License is procured to cover the external users' access to the server applications. A CAL provides the access rights to physical, virtual, and (sometimes) online services. Most of the Microsoft server products, including Windows Server, SQL Server, Exchange Server, SharePoint Server, and Microsoft Skype for Business Server (previously Microsoft Lync Server) require a CAL for each accessing user or device. With detailed information about CALs and CAL management, this chapter may help you in managing CALs through FlexNet Manager Suite.

CAL Types

A single (User or Device) CAL covers any number of servers for that product. For example, a user with a User CAL for Microsoft SharePoint Server can access any number of Microsoft SharePoint servers within the organization. Some Microsoft server products include client access rights in the server license itself. For products such as SQL Server, instead of buying CALs, you may choose to buy processor-based or core-based licenses for the accessed server applications instead of licensing under the server/CAL licensing model. Certain products like Microsoft SQL Server can be licensed based on the number of processors or cores in the host server.

The licensing rules for CALs differ depending on whether the accessing users are employees of the licensee organization or are external users accessing the licensed servers. Some versions of Microsoft SharePoint Server, such as SharePoint 2013 and SharePoint 2016 require CALs for employee access to those servers, but include rights for external users accessing those same servers. For example, a website powered by an external web server could be accessed by a possibly-unpredictable number of users and should be licensed via a version of SharePoint that includes external access rights or, for earlier versions, an External Connector License is required in addition to the server license. You may wish to check the usage required for servers in your environment and evaluate the various license models available for those servers before making your purchase decision.

For Microsoft servers that have underlying server system requirements, the relevant CALs are required for each server product accessed. For example, Microsoft SharePoint Server installations require both Microsoft Windows Server and Microsoft SQL Server. As such, each accessing user or device must have a Microsoft SharePoint CAL, Microsoft Windows Server CAL and Microsoft SQL Server CAL to comply with license requirements, unless access

is provided by the server license itself (as is the case of servers licensed per core or covered by an External Connector License).

CAL types

This section outlines the different license types used in the context of CALs. FlexNet Manager Suite only supports Microsoft User and Microsoft Device CALs via the `FlexNet Manager for Microsoft` product.

License Type	Description
User CAL	An alternative to a Device CAL, a User CAL is required per user per server product that requires a CAL. A user with a User CAL for a server product can access any instance of that server product within the enterprise via any number of devices. For example, if a user `Peter` has a single User CAL for Microsoft SQL Server product, he can access any installation of Microsoft SQL within the enterprise.
Device CAL	An alternative to a User CAL, a Device CAL is required per device per server product that requires a CAL. An accessing device with a Device CAL for a server product can be used by any number of users to access any instance of that server product. For example, if you have a Microsoft Device CAL for `Microsoft SharePoint Server` for an inventory device, any number of users can access any instance of `Microsoft SharePoint` from this device.

Other related license types

The following related license types are not directly supported by FlexNet Manager Suite. To manage any of these licenses, creation of custom license is required.

- **Client Management License (CML)**: Required per management server (SCCM) that is managing devices with non-server operating system (for example, Windows 10). If you procure device CALs for those devices that access the server, you do not require a CML.
- **External Connector License**: Required per server for each server product that is to be accessed by external users (non-employees) from outside the company network. For example, if some users of an external organization have been granted access to your `Microsoft Exchange Server`, that server requires an External Connector license.
- **Additive CALs**: A CAL required for a specific server, in addition to a User or Device CAL. Additive CAL is required per server when some advanced server functionality is accessed, in addition to the base server functionality. For example, `Windows Server Rights Management Services (RMS) CAL` is required in addition to Windows Server User or Device CAL.

Selecting a CAL Type

You can buy individual CALs for each accessing user or accessing device, or you may choose to buy CAL Suites (for a user or device). To make an effective decision about which CALs you should buy, you may want to know the usage details for the server products that require CALs. Organizations typically use a mixture of CAL licensing types based on their usage needs and the profiles of their user population.

Choosing Between User and Device CALs

If you choose to license servers via the Server/CAL license model, you may choose from user-based or device-based licenses. The **User CAL** approach requires you to purchase a CAL for every user that accesses a Microsoft server product (such as `Microsoft SharePoint Server`), regardless of the number of devices used to access that particular product. This enables users to access the services from multiple client devices, including any computers from outside the organization. The User CAL also covers access via mobile devices, such as users checking their email via smartphones or tablets. One User CAL for a server product enables you to access any number of instances of that server product. For example, if you have purchased a User CAL for Windows Server for a user, that user can access multiple Windows Servers.

Alternatively, the **Device CAL** approach requires you to purchase a CAL for every device through which the services of a server product are accessed. With an appropriate Device CAL for a device, any number of users can access the server product through that accessing device. This is often a desirable option in environments such as call centers, retail outlets or manufacturing sites in which multiple users share a single PC or kiosk. The ability to mix Device and User CALs in a single environment depends on your license agreement terms, but you must assign individual CALs to either a device or a user.

To clarify this concept, take an example of an organization with 600 users accessing Windows and Exchange servers. These 600 users work in three shifts of 200 at any time. The organization may choose to purchase 600 User CALs for Windows and 600 User CALs for Microsoft Exchange Server. However, a simple analysis shows that at any time, only 200 users are using the same computers to access the servers. Therefore, the organization is likely to buy 200 Device CALS for Exchange Server and 200 Device CALs for Windows. However, if these same 200 users are also accessing servers via their smartphones, tablets or home computers, you may find the User CAL option to be more affordable.

Note: *Microsoft typically recommends you not to mix User and Device CALs.*

For external users who connect to your organization's computers (for example, guest users or business partners), you can buy additional User CALs for each accessed server or an External Connector License, if appropriate for the quantity of external users involved.

Choosing Between Individual CALs and CAL Suites

A CAL Suite is a special license prodviding CALs for several different server products. This may make a CAL Suite more cost effective than buying an individual CAL for each distinct server product. For example, if 100 users in your organization have access to Microsoft SharePoint, Microsoft Exchange, Microsoft Skype for Business (previously Microsoft Lync Server), and Microsoft System Center Servers, it is more cost effective to purchase 100 Core CAL Suite licenses instead of 100 standalone product CALs for each of these servers. Microsoft has the following two Server CAL Suite offerings:

- Microsoft Core CAL Suite
- Microsoft Enterprise CAL Suite.

Consider the following points before you purchase CAL Suites:

- You can license these suites on per user or per device basis, and cannot split between users and devices.
- CAL Suites can only be purchased with Software Assurance (maintenance) coverage. If you do not renew the Software Assurance, the CALs are locked into the current version of each CAL available at the expiry date.

- As CAL Suites contain licenses for independently-released products, CAL Suites are version-less. CAL Suites provide the right to use the most recent version of every product in the suite.

Tip: *For detailed information about Microsoft CAL Suites, see the Microsoft website.*

Virtualization and CALs

As CAL consumption is based on the access to a server product, it does not matter whether the server product has been installed on a virtual or a physical machine. You are required to buy CALs based on the number or users or devices accessing the server software. A User CAL grants access to any instance of that server product within the enterprise. Similarly, when you buy a Device CAL for a server product, any user can access any instance of the server product through that device.

How Does FlexNet Manager Suite Calculate CAL Compliance

To determine an installation of a server application on an inventory device, FlexNet Manager Suite uses evidence (any combination of installer, file, or WMI). An additional evidence type called access evidence is separately used to measure access to the server applications, such as `Microsoft SharePoint Server`.

Note: *You need the `FlexNet Manager for Microsoft` product to create and manage CALs in FlexNet Manager Suite. To gather CAL usage inventory, you must set* ***CAL inventory options*** *while creating an inventory target to `Allow CAL access evidence collection on these targets`. For more information, see* ***Creating a Target*** *in the online help.*

When linked to an application, the access evidence records any access event to a particular server application such as `Microsoft SharePoint Server 2013`. An access evidence record provides information about the accessed application, accessing user, and accessing device. The access evidence information is collected in a separate file (`.swacc`) as a part of the discovery and inventory process. This file is only generated for the inventory devices that have the supported server products installed on Windows Server 2012 or later and for which Microsoft's User Access Logging (UAL) capability has been enabled. Access evidence is collected through various services including User Access Logging (UAL) and PowerShell scripts for some products. FlexNet Manager Suite can collect the access evidence through any of the following methods:

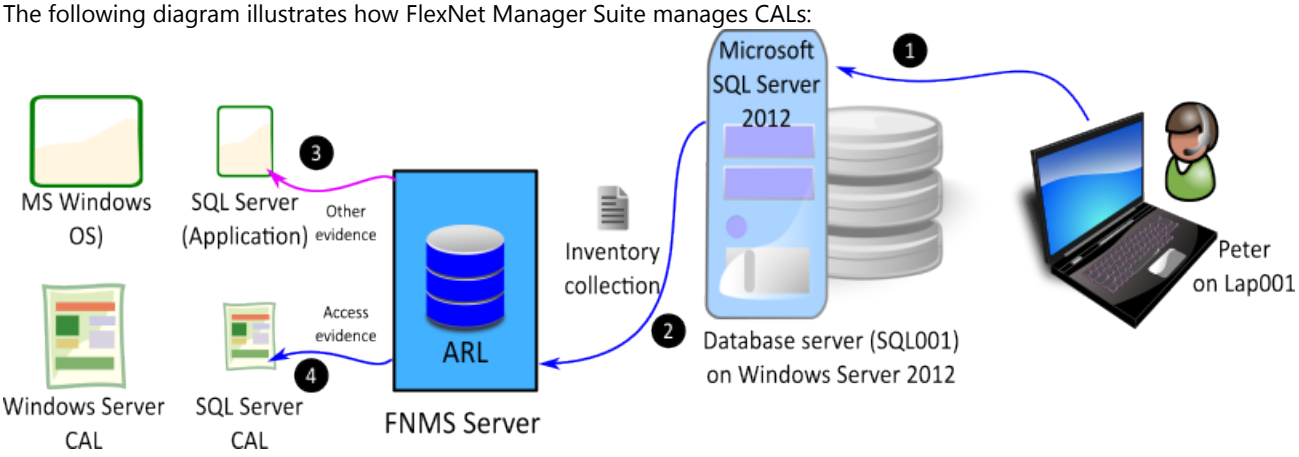
- The locally-installed FlexNet inventory agent
- Zero touch inventory collection through inventory beacon(s)
- Inventory uploads from SCCM (CAL usage inventory only for SCCM)
- Manual upload of access evidence through the **Inventory Data One-Off Upload** page

The inventory collected from any of these sources is transferred from an inventory beacon to the central application server through regularly scheduled uploads.

For conventional device licenses, during the reconciliation process the Application Recognition Library (ARL) matches the available evidence to record an application installation. For CALs, while access evidence is linked to an application, it is not used to recognize the linked application installation; instead, it is used to identify access

to that application. Like other product licenses, a CAL is also linked to the appropriate application record. However, in the case of CALs, the compliance position is generated based on the usage data (how many clients accessed the server application). The default license priorities are used to consume the appropriate license entitlement against a piece of access evidence. The collected access evidence records are visible on the **All Evidence > Access evidence** page.

The following diagram illustrates how FlexNet Manager Suite manages CALs:



1. A user `Peter` accesses the `Microsoft SQL Server 2012` installation on `SQL001` database server, installed on Microsoft Windows 2012. The access event is recorded for Microsoft Windows and Microsoft SQL Server.
2. During the next inventory collection process, FlexNet Manager Suite gathers hardware, software, and access inventory for the `SQL001` inventory device.
3. During reconciliation, FlexNet Manager Suite uses evidence to report an installation of Microsoft SQL Server 2012 and Microsoft Windows 2012.
4. The collected access evidence identifies that `Peter` accessed `Microsoft Windows Server 2012` and `Microsoft SQL Server 2012` on `SQL001`.
5. Assuming that neither `Peter` nor `Lap001` has already consumed a CAL for accessing any other Windows server, FlexNet Manager Suite consumes an entitlement of Microsoft User or Device CAL for each of Microsoft SQL Server and Microsoft Windows based on the set license priorities. For more details on license priorities, see **How Does License Consumption Order Work?** in the online help.

Variations to the above example

- If you have a per-processor or per-core license for Microsoft SQL Server on `SQL001`, CALs are not required.
- If `Peter` is an outside user (for example, a consultant on a customer site), an External Connector license may also be required.
- If this Microsoft Server Installation is used by both internal and external users, you need CALs and an External Connector license, or you can buy a Processor or Core-based license of `Microsoft SQL Server`.
  If the collected access evidence does not match a known server product, it is preserved as unrecognized evidence (visible on the **Unrecognized Evidence** page). By default, FlexNet Manager Suite retains the CAL usage inventory data for 90 days from the date of inventory collection, unless the value of the **Number of days to keep CAL usage inventory** control is changed. For details, see the **System Settings > Inventory tab** in the online help.

Access evidence collection methods

FlexNet Manager Suite supports multiple Microsoft Server applications for CAL management. Different underlying methods are used to collect access evidence data for different versions of these products. Here is a list of access evidence collection methods used to identify access to a server application:

- **User Access Logging (UAL):** When this access method is used, FlexNet Manager Suite collects the access evidence through the User Access Logging (UAL) service of Windows Server, enabled by default in Windows Server 2012 or later. This service records the client device and user request events used to access Windows Server and other installed Microsoft server applications (such as `Microsoft SharePoint Server`) into a local database. FlexNet Manager Suite collects the access evidence data through a PowerShell script that uses specific Microsoft APIs for UAL. The access events are recorded for both physical and virtual clients and servers. To use this service for a particular Microsoft server application, the UAL service should be enabled on the host Windows Server. The accessed server application should be registered with the UAL service (as happens automatically during installation). For more information about enabling the UAL service, see Microsoft documentation for the `Get-Service UALSVC` Powershell cmdlet. For more information about products registered with UAL, see Microsoft documentation for the `Get-UALOverview` PowerShell cmdlet. The appropriate inventory beacon automatically runs specific scripts to determine which user accessed which server over the last 90 days, and through which inventory device. These scripts are run as a part of the discovery and inventory process.
- **PowerShell**: When this access method is used (for Microsoft Exchange and Microsoft Skype for Business), FlexNet Manager Suite collects access data using a PowerShell script that uses specific Microsoft APIs to track the server applications accessed by users. This data is collected as a part of the discovery and inventory process. The following details are not imported when this method is used:
  - Accessing device
  - Access date
  - Access count

  Note: *FlexNet Manager Suite only gets the accessing user details (who accessed the server product) and not the accessing device details (the device through which the server product was accessed). These details are specific only to the date and time when inventory was collected, and not for the last 90 days. Therefore, Device CALs are not supported for this method. Also, when a supported server application (such as Microsoft Exchange, SharePoint, or Skype for Business) is installed on versions of Microsoft Windows Server prior to Microsoft Windows Server 2012 (for example, Microsoft Windows Server 2008 R2, Windows Server 2008, Windows Server 2003 and earlier), the server application will not have UAL support. In these instances, for each access to the supported server application, FlexNet Manager Suite creates an access evidence record for Microsoft Windows Server too. This is because a User or Device CAL for Microsoft Windows is also required, in addition to a User or Device CAL for the accessed server application.*
- **SCCM Adapter**: The SCCM adapter has been enhanced to collect the access evidence for SCCM accesses. The operator can create the corresponding core CAL license in FlexNet Manager Suite.
- **Spreadsheet imports**: You can download the CAL usage inventory template, populate it with evidence data, and upload it to create access evidence records. This method is recommended when you cannot collect access evidence. For example, when UAL service is disabled, or for servers running versions of Microsoft Windows Server prior to Microsoft Windows Server 2008 (with no UAL support). You can also use the `CAL Legacy`

license type for manual management of CALs. For more information about spreadsheet imports, see the *Importing Inventory Spreadsheets and CSV Files* chapter of this guide.

CAL consumption mode

The consumption of CALs is calculated based on the consumption mode set in the **Use rights & roles** tab of the license properties for each CAL. You can select one of the following two options:

- **Consume entitlements based on Access**: If you set the consumption mode to `Access`, FlexNet Manager Suite consumes a User CAL for all the users listed on the **All Users** page, or a Device CAL for all the devices listed on the **All Inventory** page, unless any restrictions are applied through the **Restrictions** tab of license properties. If you restrict the scope of this license to a specific enterprise group, a CAL is consumed for each user (in case of User CALs) or each device (in case of Device CALs) that are members of that enterprise group. For more information on license consumption restrictions, see the online help.

  Note: *No access evidence is required when entitlements are consumed based on access.*

- **Consume entitlements based on Usage within the time limit**: If you set the consumption mode to `Usage`, FlexNet Manager Suite consumes User or Device CAL for users or devices whose access to a server application has been reported by the appropriate access evidence. The usage tracking must be enabled for the accessed application.

Supported server products for CAL management

The following table outlines the access evidence collection methods and CAL consumption modes supported for the named Microsoft Server applications.

Table 6: Microsoft server products supported for CAL management

Microsoft Product	Product Version	User CALs Consumption		Device CALs Consumption		Inventory Data Source		Notes
		Access based	Usage based	Access based	Usage based	User CAL	Device CAL	
Windows Server	2012	Full	Full	Full	Partial	UAL	UAL	
	2012 R2	Full	Full	Full	Partial	UAL	UAL	
	2016	Full	See notes	Full	See notes			To be supported once Windows Server 2016 is released
SQL Server	2012	Full	Full	Full	Partial	UAL	UAL	Supported only on Windows Server 2012 or later
	2014	Full	Full	Full	Partial	UAL	UAL	Supported only on Windows Server 2012 or later

Microsoft Product	Product Version	User CALs Consumption		Device CALs Consumption		Inventory Data Source		Notes
		Access based	Usage based	Access based	Usage based	User CAL	Device CAL	
	2016	Full	Full	Full	Partial	UAL	UAL	Supported only on Windows Server 2012 or later
Exchange Server	2010	Full	Partial	Full	Limited	PowerShell	Spreadsheet Import	
	2013	Full	Partial	Full	Limited	PowerShell	Spreadsheet Import	
	2016	Full	Partial	Full	Limited	PowerShell	Spreadsheet Import	
SharePoint Server	2010	Full	Limited	Full	See notes	Spreadsheet Import	Spreadsheet Import	Partial support to be added later
	2013	Full	Full	Full	Partial	UAL	UAL	Supported only on Windows Server 2012 or later
	2016	Full	Full	Full	Partial	UAL	UAL	Supported only on Windows Server 2012 or later
Skype for Business (Lync)	2010	Full	Partial	Full	Limited	PowerShell	Spreadsheet Import	
	2013	Full	Partial	Full	Limited	PowerShell	Spreadsheet Import	
	2015	Full	Partial	Full	Limited	PowerShell	Spreadsheet Import	
SCCM	2012	Full	Full	Full	Full	SCCM	SCCM	Requires SCCM inventory
	2012 R2	Full	Full	Full	Full	SCCM	SCCM	Requires SCCM inventory
	2016	Full	Full	Full	Full	SCCM	SCCM	Requires SCCM inventory

Where:

- Full: Full support (UAL user data or SCCM data)
- Partial: Partial support (PowerShell script data or UAL device data)
- Limited: Limited support (requires spreadsheet import of access or usage data)

Note: *FlexNet Manager Suite collects the access evidence from the target Microsoft Windows Server, Microsoft SQL Server, and Microsoft SharePoint Server only if the* `UAL` *service of Microsoft Windows has been enabled on each accessed server. The accessed server application should be registered with the UAL service (which happens automatically during installation).*

Application usage versus CAL usage

It is important to understand the difference between application usage and CAL usage. In the above diagram, Microsoft SQL Server 2012 is an application installed on the server and recognized by FlexNet Manager Suite through imported evidence. When this application is accessed by a user (such as the database administrator) or another application installed on SQL001 device, the application usage is recorded for Microsoft SQL Server. The usage record is linked to the inventory device record of the physical server where the database has been installed.

When the service of this instance of Microsoft SQL Server is requested through a client device (for example, `Lap001`), the access event is recorded (by Microsoft UAL service) and collected for Microsoft SQL Server as CAL usage. The same process is used for Microsoft Windows 2012 on `SQL001`.

How to Manage CALs with FlexNet Manager Suite

You should undertake the following steps to manage CALs for Microsoft server applications supported by FlexNet Manager Suite:

1. Ensure that the inventory import is complete and FlexNet Manager Suite contains evidence data (including access evidence) for the devices installed with the supported Microsoft server applications that require CALs. You can verify this from the **All inventory** and **All evidence** pages.
2. Generate the **CAL Usage Inventory Report** to ensure that the access evidence has been collected.
3. Identify any server applications accessed without a CAL, by navigating to **License Compliance > Unlicensed CAL Usage**. Available only after a compliance calculation, this page displays a list of server products that have been accessed without a valid CAL. For each accessed application, you can click the number mentioned in the **Accessing devices** or **Accessing users** column to access the **CAL Usage Inventory Report** and view the related access evidence records. It is important to note that the numbers indicate the distinct users and devices, but the report displays all access evidence records.
4. On the **Access evidence** tab of the **All evidence** page, ensure that each access evidence is linked to the appropriate application. A null value for the **Name** column indicates that the evidence is not linked to the appropriate application. In such a case, link the evidence to the appropriate application. For more information, see the online help topics under **Evidence**.
5. Use the **All Licenses** page to check the usage of the existing CALs. You can set a filter `License type= Microsoft User CAL` or `License type= Microsoft Device CAL`, and check the value of the **Consumed** and **Used** columns.
6. Use the **CAL License Summary** report to view the consumption and compliance status for licenses of the type `Microsoft User CAL` and `Microsoft Device CAL`. For detailed assignment information, you may

check the **Consumption** tab in the license properties of a particular license. Note that some users may be accessing servers from their mobile devices.

7. For every CAL, ensure that you have selected the correct CAL consumption mode value through the **Consume entitlements based on** field on the **Use rights and rules** tab of the license properties. No access evidence is required if this field is set to `Access`.

8. Ensure that you have enough CALs or CAL Suite licenses to cover the usage. If you have some unused CALs, you can assign them to other users. You can use any of the following methods to create licenses:

   - **Processing purchases**: Use the **Unprocessed Purchases** page to recalculate and accept the recommendations to automatically create and link CAL purchases to the recommended CALs. For more information, see **Unprocessed Purchases** in the online help.
   - **Manual creation and linking**: You can create a license manually and link it to the appropriate server application. For more information, see **Creating a License** in the online help.

   Note: *The CAL Legacy license type is still supported for backward compatibility. You can use this license type to manually manage CALs when the access evidence cannot be collected for any reason.*

9. If required, configure the use rights for each CAL license, as well as for every product of a CAL Suite license. See the **Use Rights & Rules Tab** in the online help.

10. If required, configure the required exemptions and use rights and rules on CALs. See **License Properties > Use rights & rules** and **Consumption** tabs in the online help.

11. If required, you may allocate users and devices to appropriate CALs. Consider the following example use cases:

    - For a particular access event, you want a user to consume a CAL Suite license instead of a single-product CAL.
    - You need to exempt a user from consuming a CAL. To exempt, you need to allocate first.

    See **License Properties > Consumption tab** in the online help. Note that in the case of CALs, allocations always consume entitlements.

    Tip: *In usage scenarios where many external application users use a single server account to access a server, the* ***Allocated points*** *column value (on the* ***Consumption*** *tab of license properties) should be adjusted to show the actual consumption. For example, if multiple external-product users are using the same Microsoft SQL Server user to access the database, the UAL data will reveal access evidence for only one user of Microsoft SQL Server. In such cases, the usage should be adjusted to actual number of users to remain license compliant. The value of the* ***Allocated points*** *column should be edited to reflect the actual usage.*

12. When consuming CALs in the Access mode, you can apply restrictions to ensure that the CALs are consumed by appropriate users or devices. For more information, see **License Properties > Restrictions tab** in the online help.

13. You may use the following reports to review the CAL usage in your organization:

    - CAL License Summary
    - Licenses Consumed and Allocated by Enterprise Groups

Example Use Cases for CAL Management

This topic highlights some example scenarios for CAL management, and describes how each one of them should be managed using FlexNet Manager Suite.

Restricting CAL consumption to a specific group

You can restrict the consumption of a particular CAL to a specific enterprise group. For example, an installation of Microsoft SharePoint should only be accessed by users from the `Support` group. The following are the steps to implement this scenario using FlexNet Manager Suite:

1. Create a license of the type `Microsoft User CAL` for Microsoft SharePoint Server.
2. Ensure that all the required users have been added to the `Support` corporate group.
3. In the **Restrictions** tab of the license properties, search and add the `Support` group.
4. After the reconciliation is complete, the Microsoft SharePoint User CAL should be consumed only for the users of the added group.

Managing User and Device CALs together

Your CAL procurement strategy may include a mix of User and Device CALs. Consider the following example scenario:

- Out of the 150 users of Microsoft Exchange, 100 engineering users are full time, and work from dedicated devices.
- The remaining 50 users are in the Support department and work in two shifts, sharing 25 computers.

The following are the steps to implement this scenario using FlexNet Manager Suite:

1. Create two corporate units: `Engineering` and `Support`.
2. Create a license of the type `Microsoft User CAL` for the accessed Microsoft Exchange Server.
3. Ensure that all 100 full-time users have been added to the `Engineering` group.
4. In the **Restrictions** tab of the license properties of the `Microsoft User CAL`, search and add the `Engineering` group. The license will now be consumed only for the users of the added group.
5. Create a license of the type `Microsoft Device CAL` for the accessed Microsoft Exchange Server.
6. Ensure that all 25 support computers have been added to the `Support` group.
7. In the **Restrictions** tab of the license properties of the `Microsoft Device CAL`, search and add the `Support` group. The license will now be consumed only for the users of the `Support` group.

Note: *If the Support department users are also allowed to access the Microsoft Exchange Server via their mobile devices, home PCs, or laptops, Microsoft User CALs may be a more cost effective option to consider.*

Managing CALs for accesses from unknown external devices

The device details may not be available when Microsoft Windows Server are accessed by external users (such as consultants away on customer sites). The access evidence records will just show the IP address of the accessing device. You can adjust the CAL consumption to stay license compliant in such cases. The following are the steps to manage CALs in this scenario:

1. Navigate to the **Unlicensed CAL Usage** page and note the number in the **Accessing devices** column for the accessed product (such as Microsoft Windows Server).
2. Go to the **All Licenses** page and search the CAL for that product.
3. Open the license properties of the license and click the **Consumption** tab.
4. Add the number mentioned in the **Accessing devices** column on the **Unlicensed CAL Usage** page to the count mentioned in the **Allocated point** for this license. This will adjust the number of unknown accesses being made to the server application.

Managing CALs for multiple indirect accesses to a server application

Multiple users of a web application or website may access a server application through the same account. For example, 100 users of an ERP solution may access Microsoft SQL Server through an internally-configured `ErpQuery` account. As only one user-account of SQL server is accessing the application, the access evidence will only indicate access by one user, instead of reporting access by 100 users. To be license compliant, you should consume 100 User CALs for Microsoft SQL Server. The following are the steps to manage CALs in this scenario:

1. Check the CAL Usage Inventory report to find a very high value in the **Access Count** column in an access record. This would provide a hint that multiple users are accessing a server application.
2. Using any method to record the number of indirect accesses, make a note of the number of users of the ERP application that are indirectly accessing the Microsoft SQL Server installation.
3. Go to the **All Licenses** page and search the appropriate User CAL license for Microsoft SQL Server.
4. Open the license properties of the license and click the **Consumption** tab.
5. Add the number noted in step 1 to the count mentioned in the **Allocated point** for this license. This will adjust the consumption for the accesses being made to the server application.

Note: *The license requirements would be different In the case of unknown external users or devices accessing an Internet-facing site that is running on Microsoft SharePoint Server. If Microsoft SharePoint version 2013 or 2016 is being used, the server license itself covers unlimited external accesses, and no additional CALs are required. For Microsoft SharePoint version 2010 and earlier, you need an External Connector Licenses to cover the anonymous access.*

Appendix A- Template Details for CAL Usage Inventory Upload

You can import CAL usage inventory through spreadsheet imports when you cannot collect access evidence for the accessed server applications. You can schedule regular inventory imports through the FlexNet Beacon, or you can use the Inventory Data One-Off Uploads feature (see *Data Inputs* in the online help) to upload CAL usage inventory spreadsheet. For more information on inventory uploads, see *Importing Inventory Spreadsheets and CSV Files*.

To upload CAL usage inventory, the spreadsheet template should be populated with at least all mandatory (**Required**) columns, and uploaded to FlexNet Manager Suite. The following table describes the columns of the CAL Usage Inventory spreadsheet template, accessible through the Inventory **Data One-Off Upload** page:

Column	Description	Mandatory
AccessCount	The number of times that the server application was accessed.	No
AccessDate	The date on which the server application was accessed.	No
AccessingDeviceComputerName	The **ComputerName** of the accessing device (through which the server application was accessed).	No, if **AccessingUser** is specified.
AccessingDeviceDomain	The domain name of the accessing device.	No
AccessingDeviceIPAddress	The IP address of the accessing device.	No, if **AccessingUser** is specified.
AccessingDeviceSerialNo	The serial number of the accessing device.	No
AccessingUser	The `DOMAIN/SAMAccountName` for the accessing user.	No, if **AccessingDeviceIPAddress** or **AccessingDeviceComputerName** is specified.
ComputerID	The ComputerID of the accessed device (where the server application has been installed). It must match the **ComputerID** specified for this device in the **Computer** spreadsheet, or the row will be ignored. Uploading the Computer spreadsheet is mandatory with the CAL inventory upload.	Yes
Edition	The edition of the accessed server application as reported by the access evidence. This data is not used for access recognition.	No

Column	Description	Mandatory
InventoryDate	Inventory date of the evidence. If not provided, defaults to the current Coordinated Universal Time (UTC).	No
ProductName	The product name of the accessed application as reported by the access evidence. For all supported server applications (except SCCM), the Version and Edition (if present) is included in the ProductName. For example, `Microsoft SharePoint Server 2013 Enterprise Preview`	Yes
Version	The version of the accessed server application as reported by the access evidence. ***Note:*** *This column should be populated only for SCCM.*	No

4

Introduction to Microsoft Office 365

Microsoft Office 365 is a set of productivity tools available online (through a cloud-based subscription method) and offline (installed locally). Microsoft Office 365 subscriptions include a range of productivity tools from Microsoft Office, Microsoft Exchange, Microsoft SharePoint, Microsoft Yammer, and Microsoft Skype for Business product lines. Multiple subscription plans (such as E1, E3, and E5) are available. Some of these plans allow you to install Office applications locally.

With detailed information about Microsoft Office 365, this chapter may help you in managing Microsoft Office 365 licenses through FlexNet Manager Suite.

Microsoft Office 365 License Management Considerations

Microsoft Office 365 licenses that allow local Office installations, also allow you to install Microsoft Office 365 on up to five PCs or Macs, and up to five tablets (subject to the chosen plan) per `Named User` license. The following section provides an overview of the tools provided in each enterprise plan. With the *FlexNet Manager for Microsoft* product, you can use FlexNet Manager Suite to manage your Microsoft Office 365 subscriptions. You can:

- Validate that the desired users have the right plan
- Determine how many Office 365 licenses you need to purchase while negotiating license agreements
- Ensure that business units are not under or over-subscribed.

Based on the selected plan, a subscription of Microsoft Office 365 may allow you to access multiple Office applications online as well as offline (local installations) on up to five computers, and five tables or smart phones. Due to this change, the compliance calculations for Microsoft Office 365 licenses are different from other applications.

For example, the *E3* plan allows you to install and use the following applications:

- Exchange Online Plan 2 for Office 365
- Skype for Business Online for Office 365

- Office Professional Plus for Office 365
- Azure Rights Management for Office 365
- SharePoint Online for Office 365
- Yammer for Office 365

Note: *For a detailed description of the available Microsoft Office 365 subscription plans, and the applications supported in each plan, see Microsoft Office 365 website.*

Considerations for managing Microsoft Office 365 licenses

- **Mobile devices**: Because FlexNet Manager Suite does not collect inventory from all mobile devices (such as iPads), you may have to deactivate Microsoft Office 365 from one of the mobile devices when the number of devices used to author and edit Microsoft Office 365 documents exceeds five. Note that you can read or present documents on a deactivated copy of Microsoft Office 365. You can get the details of installed devices from the Office 365 Admin Center.
- **Local installations**: As Microsoft Office 365 is a `Named User` license, FlexNet Manager Suite assumes that an installation on a device owned by the primary user (who has a subscription for Office 365) is licensed. Microsoft manages the access to the Microsoft Office 365 Online Service.
- **Devices**: Though the installed device information is available to Office 365 admins through the Office 365 Admin Center, FlexNet Manager Suite imports no device information from the Microsoft Office 365 Online Service. The imported users are mapped to the compliance users, and the devices assigned to those users are considered as the accessing devices. If an imported user is not assigned with any inventory device, a dummy remote device is created by using the naming convention *`User Name`* `(Remote)`. Such a dummy device is not treated as a managed device by FlexNet Manager Suite. If an installation of Office 365 is detected on a device, and the device is not found in the Microsoft Office 365 Online Service, the installation should consume an existing perpetual license.
- **Users**: Each user imported from the Microsoft Office 365 Online Service is mapped to the appropriate compliance user record (by matching the **Email** or **Alternate email** field values). If an imported user cannot be mapped to any compliance user, a compliance user record is created with a dummy inventory device assigned to it. These users consume against the Microsoft Office 365 multi-product license created for the Microsoft Office 365 tenant.
- **Office versions**: A perpetual license of Office 365 enables you to downgrade to an earlier version but the subscription license does not. If you have a perpetual license of the older Office version, and you buy a subscription, the license validity of the older version gets extended until the subscription expiry date.
- **License Entitlements**: The entitlements data for Microsoft Office 365 is directly imported from the Microsoft Office 365 Online Service. If no license exists for Microsoft Office 365, a Named User license is created for each Microsoft Office 365 tenant (with available entitlements); otherwise, the available entitlements are added to an existing license of Microsoft Office 365. If you want to add purchases for Microsoft Office 365 (only for tracking purposes), the value of the **Assigned** column on the **Purchases** tab of the license properties is set to `0`. The **Entitlement status** displays a value `Not contributing` for any Microsoft Office 365 purchases.

Managing Office 365 Licenses through FlexNet Manager Suite

The following is the procedure to manage Microsoft Office 365 licenses through FlexNet Manager Suite:

1. Ensure that you have created and configured a connector for each tenant of the Microsoft Office 365 Online Service. For details on how to create a connector to Microsoft Office 365, see Managing Connections to Microsoft Office 365 Online Service.
2. After the next discovery and inventory collection, FlexNet Manager Suite imports the following objects:
   - **Users**: Each imported user from the Microsoft Office 365 Online Service is mapped to the appropriate compliance user record (by matching the **Email** or **Alternate email** field values). If an imported user cannot be mapped to any compliance user, a compliance user record is created with a dummy inventory device assigned to this user.
   - **Licenses**: A `Named User` multi-product license (`Microsoft Office 365`) is created with multiple linked applications for each Microsoft tenant. The name of the Microsoft tenant is prefixed to the name of each of these licenses. This helps to distinguish licenses when you have multiple Microsoft Office Online Service accounts. If a particular plan is not supported by FlexNet Manager Suite, a license with no application is created. When the support for such plan is added during subsequent automatic updates to the Application Recognition Library, a recommended change to link applications will appear.
   - **Applications**: The information about the subscribed applications is imported and a multi-product license containing these applications is created.
3. See the **All Users** and **All licenses** pages to ensure that the required users and licenses have been created.
4. After license reconciliation has been completed, the **Product Summary** page should show the compliance status for Microsoft Office 365.
5. Navigate to **License Compliance > All Licenses** and filter the records to view Microsoft Office 365 licenses. You can check the **Consumed** and **Used** columns to know how many subscriptions are being used.
6. You can also view the following reports for insights on Microsoft Office 365 license consumption:
   - `User license details`
   - `License Overlap`.

> ***Note:*** *A subscription of Microsoft Office 365 enterprise plan (except E1) allows you to install Microsoft Office 365 on up to five PCs or Macs, and up to five tablets. FlexNet Manager Suite can only track the installation on devices that are present within your enterprise network. If you wish to include any installation outside the enterprise network, you can create a dummy inventory device and allocate a Microsoft Office 365 license to track the usage.*

Changes in subsequent imports

Any subsequent imports from the Microsoft Office 365 Online Service updates the following for each license of Microsoft Office 365:

- Entitlement count.
- Expiry date.
- Sets the **Allocations consume license entitlements** check box under **License consumption rules** accordion on the **Use rights & rules** tab of the properties of the `Named User` multi-product license created for each Microsoft Office 365 Service tenant. This is because Microsoft Office 365 is a `Named User` license and allocations must consume entitlements.
- Sets the **Allocated** check box in the **Consumption** tab of the license properties.

Connecting to The Microsoft Office 365 Online Service

To generate the compliance position for Microsoft Office 365 licenses, FlexNet Manager Suite needs information about subscription and usage of Microsoft Office 365 licenses (both online and offline). The inventory collection process returns the local installations of Microsoft Office that are tied to Microsoft Office 365 subscription. To get the subscription and usage information from Microsoft Office Online Service, the FlexNet Beacon should be configured with an inventory connection of the type `Microsoft Office 365` for each tenant of Microsoft Office 365.

When created and configured with the Microsoft Office 365 connection, the inventory beacon imports the licenses, users, and usage information from the Office 365 Online Service account and uploads it to FlexNet Manager Suite according to the defined schedule. For details on how to create a connector to Microsoft Office 365, see Managing Connections to Microsoft Office 365 Online Service.

Note: *The Microsoft Office 365 support is available only with the FlexNet Manager for Microsoft product.*

Managing Connections to Microsoft Office 365 Online Service

Use the following procedure to create a connection to the Microsoft Office 365 Online Service on FlexNet Beacon. A connection is required for each tenant of the Microsoft Office 365 online service. The inventory beacon requires this connection to import entitlements, users, and usage information from the Microsoft Office 365 online account.

Ensure that the Microsoft Office 365 Online Service tenant user has the required privileges for this operation. For more information, see https://support.office.com/en-gb/article/About-Office-365-admin-roles-da585eea-f576-4f55-a1e0-87090b6aaa9d?ui=en-US&rs=en-GB&ad=GB.

Also make sure that the following prerequisites are met on each inventory beacon that needs to download data from the Microsoft Office 365 Online Service (noting that the order of installation of prerequisite software may be significant). These requirements should have been met when the inventory beacon was installed:

- You have licensed the FlexNet Manager for Microsoft product.

- The inventory beacon that will collect inventory for Office 365 in the cloud is a 64-bit machine (prerequisite software is not available for 32-bit architectures)
- Microsoft .NET framework version 4.5.2 or later is running on the inventory beacon.
- PowerShell 3.0 or later is running on Windows Server 2008 R2 SP1 or later, or Windows 7 SP1 or later; with the PowerShell execution policy set to `RemoteSigned`.
- Microsoft Online Services Sign-in Assistant for IT Professionals RTW is installed (instructions https://technet.microsoft.com/library/dn975125.aspx, direct link to the MSI https://www.microsoft.com/en-us/download/details.aspx?id=41950).
- Windows Azure Active Directory Module for Windows PowerShell is installed (instructions https://technet.microsoft.com/library/dn975125.aspx, direct link to the MSI http://go.microsoft.com/fwlink/p/?linkid=236297).
- Skype for Business Online, Windows PowerShell Module (https://www.microsoft.com/en-us/download/details.aspx?id=39366) is installed on the inventory beacon.

Tip: *The PowerShell execution policy can be correctly set with the following command:*

```
Set-ExecutionPolicy RemoteSigned
```

To create a connection to the Microsoft Office 365 Online Service:

1. Select the **Inventory Systems** page in the FlexNet Beacon interface.
2. Choose either of the following:
   - To change the settings for a previously-defined connection, select that connection from the list, and click **Edit...**.
   - To create a new connection, click the down arrow on the right of the **New** split button, and choose **Powershell**.
3. Complete (or modify) the values in the dialog that appears. All of the following values are required:

   Note: *If you have multiple subscriptions to Microsoft Office 365 (for example, separate subscriptions for different corporate units or locations), you need to create a separate connector for each subscription using its own credentials.*

   - **Connection Name**: The name of the inventory connection. When the data import through this connection is executed, the data import task name is same as the connection name.
   - **Source Type**: Select `Microsoft Office 365` from this list.
   - **Username**: The Microsoft Office 365 Online Service tenant user name.
   - **Password**: The Microsoft Office 365 Online Service tenant password.

   Tip: *The account used requires at least the following privileges in Office 365:*

   - *Member of the* ***Limited/Customized Administrator*** *role*

- ***Billing administrator*** *option selected*
- ***Skype for Business administrator*** *option selected.*

4. Save the connection.

After a successful data import, the users, applications, licenses, and usage data are all visible in the appropriate grids of FlexNet Manager Suite.

Note: *To know more about the operations available on the* ***Inventory Systems*** *page, see Inventory Systems Page. For scheduling data imports through this connection, see Scheduling a Connection.*

5

Oracle Discovery and Inventory

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 have licensed against what software you have installed. To collect detailed Oracle inventory, you must have FlexNet Manager for Oracle, a separately licensed product for FlexNet Manager Suite. Oracle discovery and inventory collection processes involve 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 license entitlements purchased.

Note: *The discovery and inventory collection works in the same way on physical and virtual machines.*

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`. 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 Oracle WebLogic) 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.

Tip: *When Oracle Enterprise Cloud Control 12c is used to manage Oracle databases, the licensing information is stored in the Oracle Management Repository on the Oracle Enterprise Cloud Control instance, instead of on each database instance. To get the complete licensing data, you need to collect inventory from the instance with Oracle Enterprise Cloud Control 12c, as well as from each database instance being managed by it. FlexNet Manager Suite automatically merges and resolves the inventory data to generate an accurate licensing position.*

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 uses the running process listings to find the SID, ORACLE_HOME, and Owner (running operating system user which is also a member of the `dba` group) to collect 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. After adoption, the FlexNet inventory agent runs as the local operating system user currently logged into the Oracle server. You can also configure the agent to run as a specific Sysdba user by setting `OracleInventoryAsSysdba` parameter of the `ndtrack` to `False`, and configuring the desired Sysdba user through the `OracleInventoryUser` parameter in the `config.ini` file. For more information, see Appendix D - Inventory Collection Through ndtrack With a Specific DBA.

Oracle LMS Hardware Collection

FlexNet Manager Suite automatically executes LMS scripts provided by Oracle in order to find and collect hardware inventory for every device with one or more Oracle databases installed on it. For information about globally disabling execution of local scripting enabled in the `InventorySettings.xml` file, refer to the *PerformLocalScripting* option in the *Gathering FlexNet Inventory* PDF file, available through the title page of the online help.

Tip: *This process requires that the FlexNet Manager for Oracle product has been licensed.*

Privileges required

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

- **For Windows**: The FlexNet inventory agent requires `administrative` privileges in order to run Oracle LMS hardware scripts.
- **For UNIX**: The FlexNet inventory agent requires `root` privileges in order to run Oracle LMS hardware scripts.

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` of FlexNet inventory agent is remotely executed on every Oracle server identified by the target definition. The `ndtrack` uses the running process listings to find the SID, ORACLE_HOME, and Owner (running operating system user which is also a member of the `dba` group) to collect hardware, software, Oracle Database, and Oracle options discovery and inventoryIt 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 as the local operating system user currently logged into the Oracle server. You can also configure the agent to run as a specific Sysdba user by setting `OracleInventoryAsSysdba` parameter of the `ndtrack` to `False`, and configuring the desired Sysdba user through the `OracleInventoryUser` parameter in the `config.ini` file.
  The `ndtrack` uses the running processes to find the SID, ORACLE_HOME, and Owner (running user), and then runs as owner to collect Oracle inventory.

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					

Feature	Agent-based	Zero touch	Direct- nw scan	Direct- tnsnames.ora	Direct- manual	Spreadsheet upload
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: • **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.			Y			
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 produce 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 product, 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, 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). You can also configure the agent to run as a specific Sysdba user by setting `OracleInventoryAsSysdba` parameter of the `ndtrack` to `False`, and configuring the desired Sysdba user through the `OracleInventoryUser` parameter in the `config.ini` file.
  - **On UNIX**: A local account is required that has `ssh` privileges. You must register this account in the secure Password Store on the appropriate inventory beacon. The `ndtrack` runs as the local operating system user currently logged into the Oracle server. You can also configure the agent to run as a specific Sysdba user by setting `OracleInventoryAsSysdba` parameter of the `ndtrack` to `False`, and configuring the desired Sysdba user through the `OracleInventoryUser` parameter in the `config.ini` file.
- 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, if `ndtrack` is running as a specific dba user, that user must be a member of the `dba` group. You can perform the following test to verify whether Oracle local OS authentication is

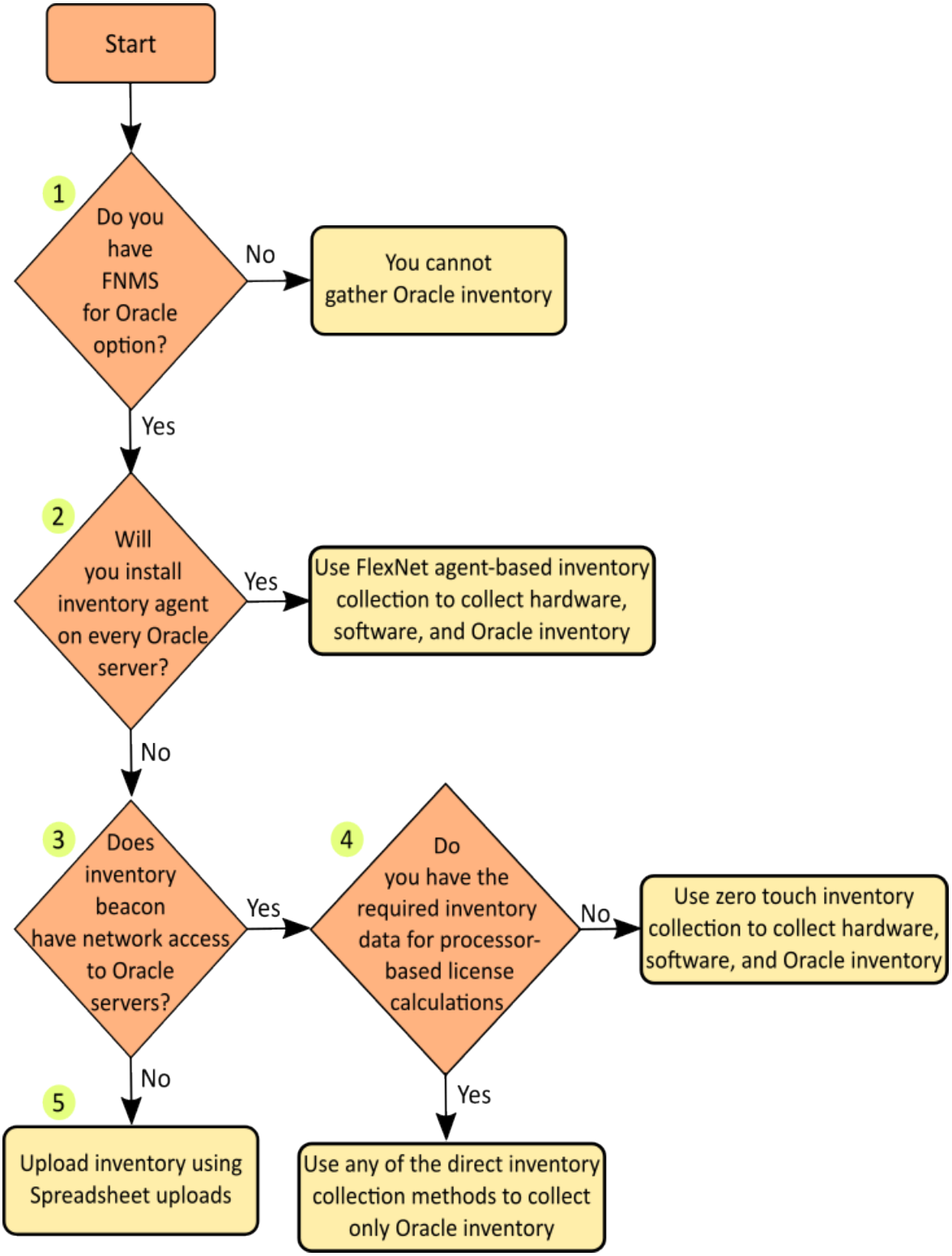
adequately configured for the FlexNet inventory agent (in either agent-based or zero touch inventory collection) to gather Oracle inventory:

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:



1. To perform license management and compliance for Oracle-specific licenses, you need a license for the FlexNet Manager for Oracle product. Without this product, 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.
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

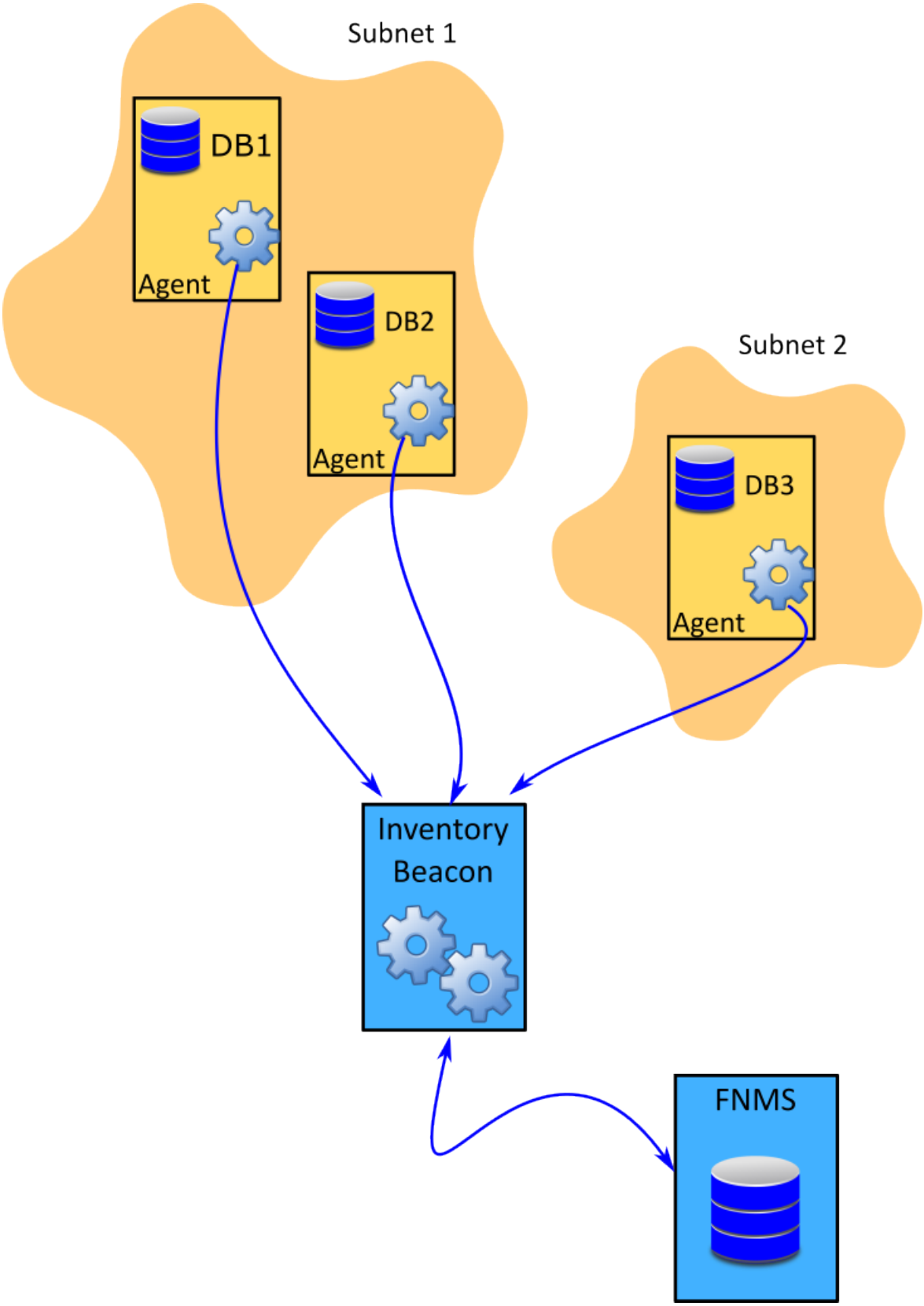
In agent-based inventory collection, FlexNet Manager Suite collects Oracle inventory through a FlexNet inventory agent installed on each Oracle server within your network. When the agent inventory collection job (scheduled from the web interface) executes, each FlexNet inventory agent gathers discovery 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.


Subnet 1
DB1
Agent
DB2
Agent
Subnet 2
DB3
Agent
Inventory Beacon
FNMS

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. If `tnsnames.ora` is not found, the `ndtrack` gets the SID for the Oracle service through the `oratab` file. 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, for the adoption process, a local account that has `ssh` privileges is required. This account must be registered this account in the secure Password Store on the appropriate inventory beacon. After adoption, the FlexNet inventory agent runs as the local operating system user currently logged into the Oracle server to collect discovery and inventory. You can also configure the agent to run as a specific Sysdba user by setting `OracleInventoryAsSysdba` parameter of the `ndtrack` to `False`, and configuring the desired Sysdba user through the `OracleInventoryUser` parameter in the `config.ini` file.

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 and Oracle Inventory Collection Methods.
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. You can also configure the agent to run as a specific Sysdba user by setting `OracleInventoryAsSysdba` parameter of the `ndtrack` to `False`, and configuring the desired Sysdba user through the `OracleInventoryUser` parameter in the `config.ini` file. 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.

     Note: *To collect local Oracle 9i database inventory on Windows, you need to run `ndtrack` with any Windows system user that is a member of the `ora_dba` group. The following command would collect Oracle inventory using a Windows user `User1` that is an administrative user and member of the `ora_dba` group.*

     ```
     ndtrack.exe -t machine <User1>
     ```

   - **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 as the local operating system user currently logged into the Oracle server. You can also configure the agent to run as a specific Sysdba user by setting `OracleInventoryAsSysdba` parameter of the `ndtrack` to `False`, and configuring the desired Sysdba user through the `OracleInventoryUser` parameter in the `config.ini` file.
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 from all target devices options in 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. If `tnsnames.ora` is not found, the `ndtrack` gets the SID for the Oracle service through the `oratab` file.

> ***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` as the local operating system user currently logged into the Oracle server.
11. The `ndtrack` uses the following procedure to collect Oracle discovery and inventory information:
    a. It checks the running process listings to discover the running instances of Oracle Database, their SIDs, and database instance owners (operating system user running Oracle Database on the Oracle instance), and subsequently collect inventory from the discovered Oracle Database instances. The locally-installed FlexNet inventory agent cannot find ORACLE_HOME for Oracle servers running AIX and Solaris 9 (or earlier), and uses file scan to find ORACLE_HOME. File scanning is the only method to discover ORACLE_HOME for Oracle servers running on HP_UX.

    b. The `ndtrack` connects to the Oracle Database as the owner (local OS user running Oracle Database) to collect inventory.

    c. If the ORACLE_HOME path cannot be determined from a running process, the `ndtrack` reads the ORACLE_HOME path from the `oratab` file and discovers ORACLE_HOME and SID for each Oracle service. This method can discover multiple Oracle instances that use different accounts.

d. If the ORACLE_HOME path cannot be determined, and the `oratab` file is not found, the `ndtrack` runs a file system scan to discover ORACLE_HOME and SID of the running Oracle instances. This scan is performed on the folders specified on the **Inventory Settings** page. For more information, see the **Inventory Settings** page in the online help.

e. If `ndtrack` fails to identify the operating system user running Oracle instance, it impersonates the first user in the `dba` group to collect Oracle inventory.

12. The FlexNet inventory agent sends the discovery (`.disco` files) and inventory (`.ndi` files) information to the appropriate inventory beacon.
13. The inventory beacon uploads this information to FlexNet Manager Suite.
14. 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.
15. 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.*

16. If the **Status** column displays `Completed with errors`, check the adoption status for errors. For more information, see Troubleshooting Oracle Discovery.
17. 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.
18. Navigate to the **Discovery & Inventory > Oracle Instances** page. You should see your Oracle servers listed on this page.

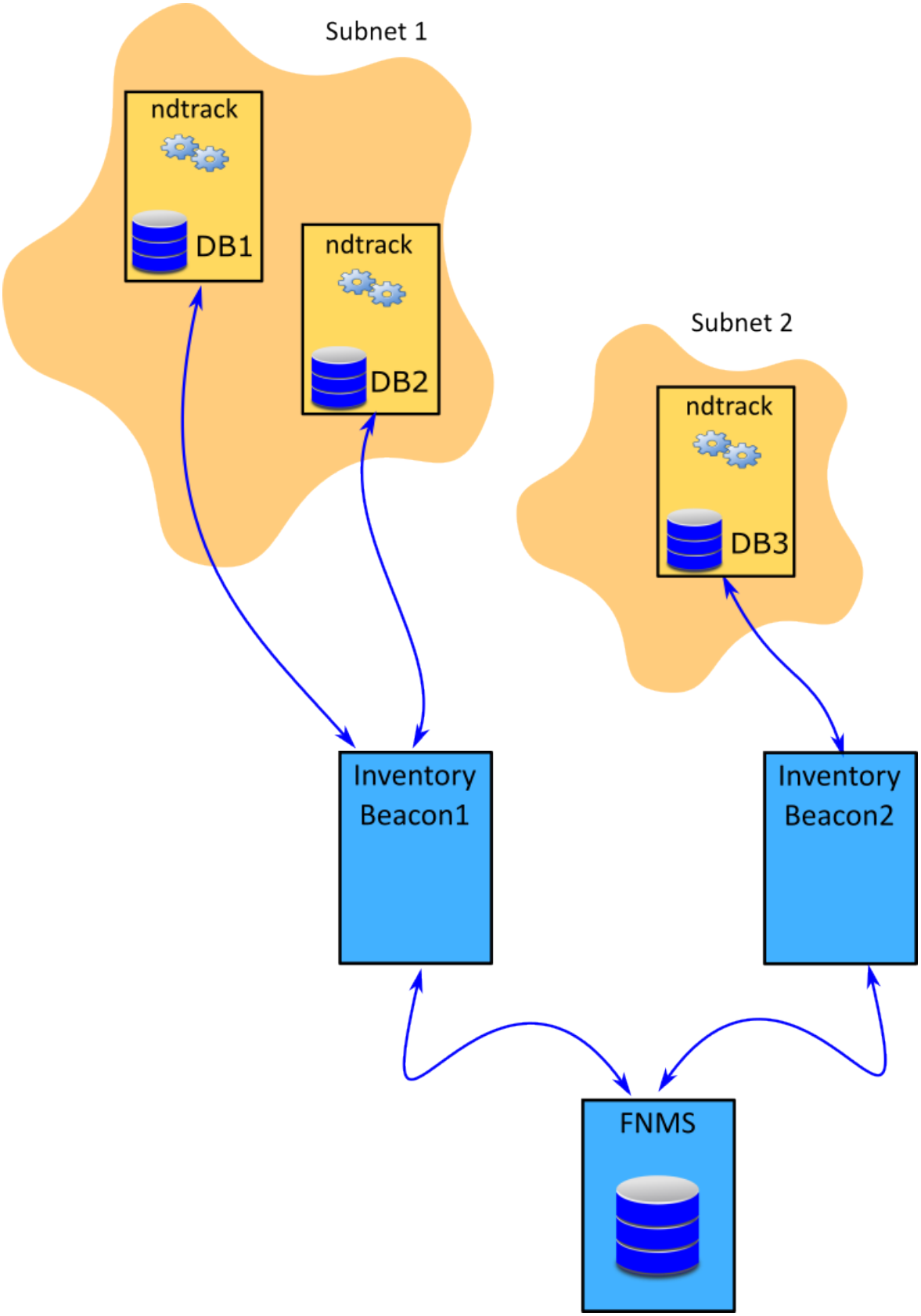
Note: *You can also collect inventory if there is no network connection between the FlexNet inventory agent and appropriate inventory beacon. See Appendix C - Inventory Collection when Inventory Beacon is Disconnected from FlexNet Agent.*

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` 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:


Subnet 1
ndtrack
DB1
ndtrack
DB2
Subnet 2
ndtrack
DB3
Inventory Beacon1
Inventory Beacon2
FNMS

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` 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 remote FlexNet inventory agent (`ndtrack`) runs as the local operating system user currently logged into the Oracle server. You can also configure the agent to run as a specific Sysdba user by setting `OracleInventoryAsSysdba` parameter of the `ndtrack` to `False`, and configuring the desired Sysdba user through the `OracleInventoryUser` parameter in the `config.ini` file. For more information, see Appendix D - Inventory Collection Through ndtrack With a Specific DBA.

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 and Oracle Inventory Collection Methods.
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`

Property	Expected Value
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 agent runs as the local operating system user currently logged into the Oracle server. You can also configure the agent to run as a specific Sysdba user by setting `OracleInventoryAsSysdba` parameter of the `ndtrack` to `False`, and configuring the desired Sysdba user through the `OracleInventoryUser` parameter in the `config.ini` file.
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` as the local operating system user currently logged into the Oracle server.

  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. The `ndtrack` uses the following procedure to collect Oracle discovery and inventory information:
    a. It checks the running process listings to discover the running instances of Oracle Database, their SIDs, and database instance owners (operating system user running Oracle Database on the Oracle instance), and subsequently collect inventory from the discovered Oracle Databases. The `ndtrack` cannot find ORACLE_HOME for Oracle servers running AIX and Solaris 9 (or earlier), and uses file scan to find ORACLE_HOME. File scanning is the only method to discover ORACLE_HOME for Oracle servers running on HP_UX.
    b. The `ndtrack` connects to the Oracle Database as the owner (local OS user running Oracle Database) to collect inventory.
    c. If the ORACLE_HOME path cannot be determined from a running process, the `ndtrack` reads the ORACLE_HOME path from the `oratab` file and discovers ORACLE_HOME and SID for each Oracle service. This method can discover multiple Oracle instances that use different accounts.
    d. If the ORACLE_HOME path cannot be determined, and the `oratab` file is not found, the `ndtrack` runs a file system scan to discover ORACLE_HOME and SID of the running Oracle instances. This scan is performed on the folders specified on the **Inventory Settings** page. For more information, see the **Inventory Settings** page in the online help.
    e. If `ndtrack` fails to identify the operating system user running Oracle instance, it impersonates the first user in the `dba` group to collect Oracle inventory.
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.
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.

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. 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.*

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:

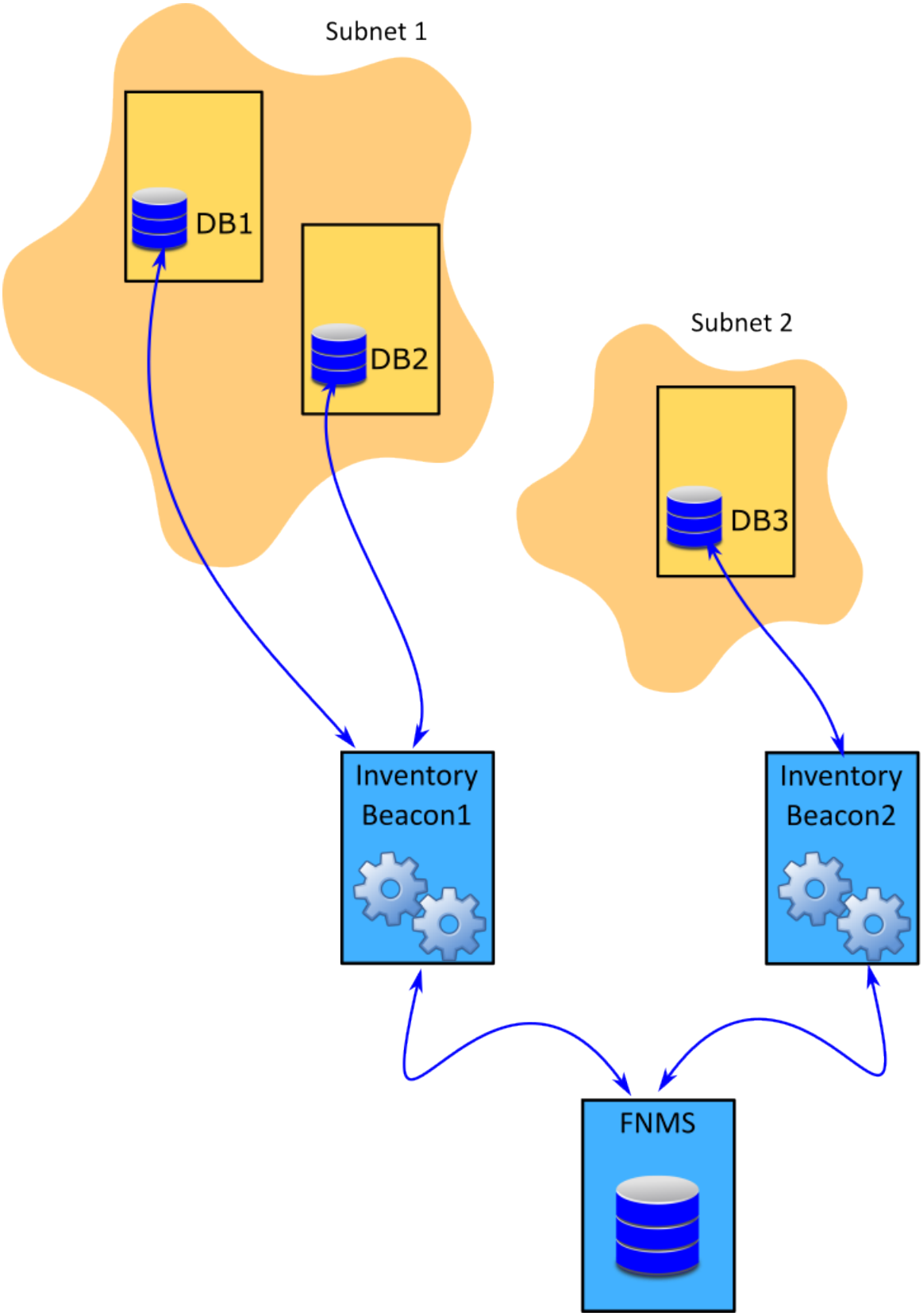
- **Port scan**: Enables FlexNet Beacon engine to connect to Oracle Database using the specified ports. FlexNet Manager Suite adds 1521 and 2483 as default ports. You can change the default ports or add more as your environment requires.
- **SNMP scan**: Enables FlexNet Manager Suite to connect to Oracle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

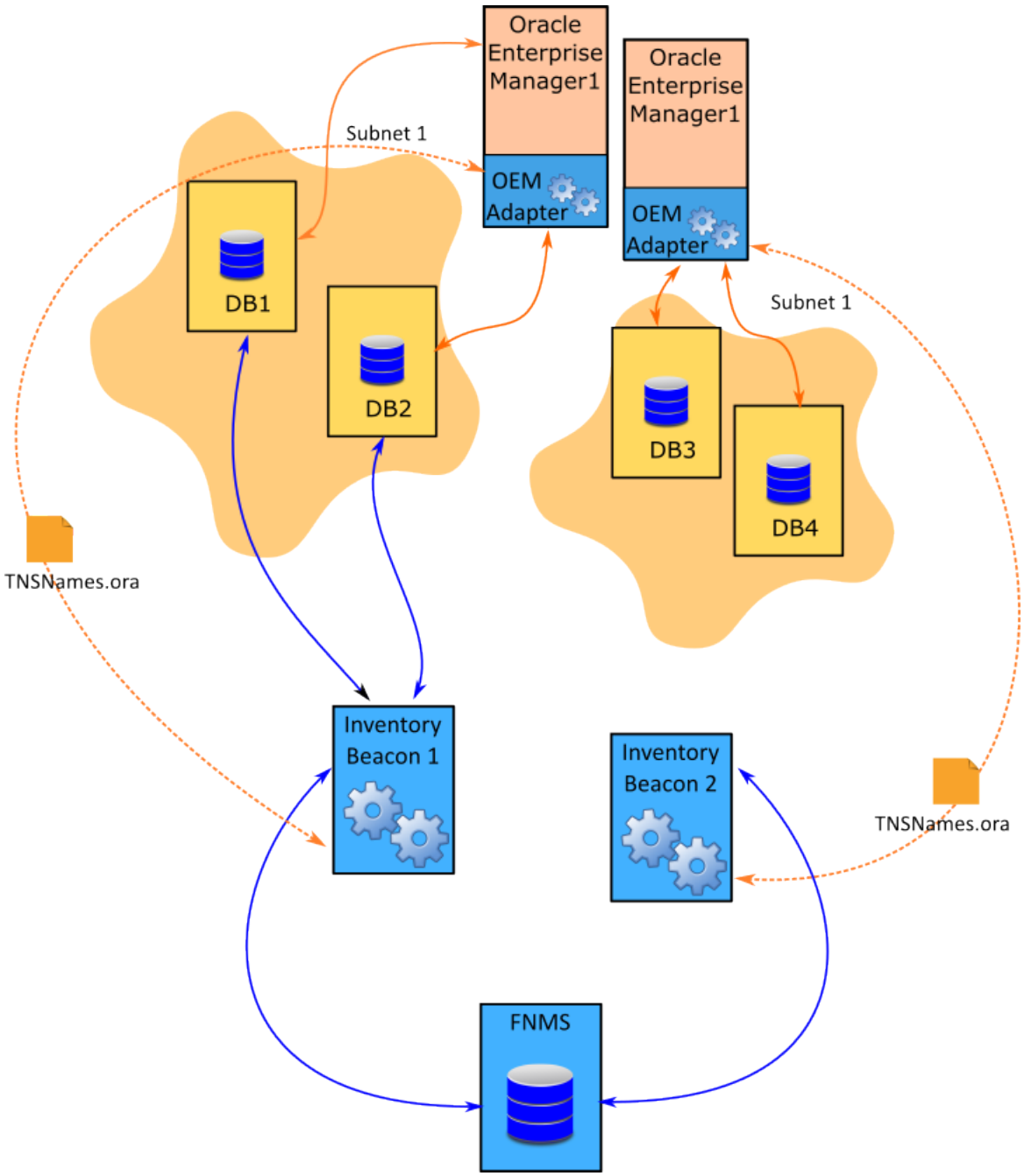
name) and the action specifies how to discover Oracle servers (for example, using the `tnsnames.ora` file) and what action to perform on them. When the rule executes, it instructs each FlexNet Beacon engine to connect to Oracle databases and collect Oracle inventory if the targeted devices are within 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:


Subnet 1
DB1
DB2
Subnet 2
DB3
Inventory Beacon1
Inventory Beacon2
FNMS

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 and Oracle Inventory Collection Methods.
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. If the Oracle listener has been configured with a password, record that password in the secure Password Store on the appropriate inventory beacon. No user name is required when you record the listener password. This setting is required for Oracle Database discovery, and is different from recording the user name and password for the inventory collection user as described in the following step.

8. 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. 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.

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 in the online help.*

9. 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.

10. 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 Method	Action Settings
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 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 from all target devices****.*
- ***Schedule****: Specify the running schedule for this rule.*

11. 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.
12. 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.
13. 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.*

14. 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.

15. 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.
16. 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.** • **Gather hardware and software inventory from all target devices**.

Inventory collection method	Required actions
Direct inventory using network scan	Select the following options in the action definition: • **Discover devices using network scan** and **Gather hardware and software inventory from all target devices** 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 from all target devices** 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 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 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 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.

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.

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 from all target devices** 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.
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 and Appendix A- Oracle Tables and Views for Oracle Inventory Collection.
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.
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.

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. 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.

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.

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
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.
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:

- `ALL_SDO_GEOM_METADATA SELECT`
- `DBA_ADVISOR_TASKS SELECT`
- `DBA_AUDIT_TRAIL SELECT`
- `DBA_AWS SELECT`
- `DBA_CPU_USAGE_STATISTICS SELECT`
- `DBA_CUBES SELECT`
- `DBA_DV_REALM SELECT`
- `DBA_ENCRYPTED_COLUMNS SELECT`
- `DBA_FEATURE_USAGE_STATISTICS SELECT`
- `DBA_FLASHBACK_ARCHIVE_TABLES SELECT`
- `DBA_FLASHBACK_ARCHIVE_TS SELECT`
- `DBA_FLASHBACK_ARCHIVE SELECT`
- `DBA_INDEXES SELECT`
- `DBA_LOB_PARTITIONS SELECT`
- `DBA_LOB_SUBPARTITIONS SELECT`
- `DBA_LOBS SELECT`
- `DBA_MINING_MODELS SELECT`

- DBA_OBJECT_TABLES SELECT
- DBA_OBJECTS SELECT
- DBA_RECYCLEBIN SELECT
- DBA_REGISTRY SELECT
- DBA_SEGMENTS SELECT
- DBA_SQL_PROFILES SELECT
- DBA_SQLSET_REFERENCES SELECT
- DBA_SQLSET SELECT
- DBA_TAB_PARTITIONS SELECT
- DBA_TAB_SUBPARTITIONS SELECT
- DBA_TABLESPACES SELECT
- DBA_TABLES SELECT
- DBA_USERS SELECT
- DUAL SELECT
- GV_$INSTANCE SELECT
- GV_$PARAMETER SELECT
- MGMT$TARGET_PROPERTIES
- MODEL$ SELECT
- REGISTRY$HISTORY SELECT
- ROLE_SYS_PRIVS SELECT
- SDO_GEOM_METADATA_TABLE SELECT
- USER_ROLE_PRIVS SELECT
- USER_SYS_PRIVS SELECT
- UTL_INADDR EXECUTE
- V_$ARCHIVE_DEST_STATUS SELECT
- V_$BLOCK_CHANGE_TRACKING SELECT
- V_$CONTAINERS SELECT
- V_$DATABASE SELECT

- V_$INSTANCE SELECT
- V_$LICENSE SELECT
- V_$OPTION SELECT
- V_$PARAMETER SELECT
- V_$SESSION SELECT
- V_$VERSION SELECT

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 *Gathering FlexNet Inventory* 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 the local operating system user currently logged into the Oracle server 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 *Gathering FlexNet Inventory* 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 Gathering FlexNet Inventory guide contains complete details about setting up the environment in the disconnected mode.*

Appendix D - Inventory Collection Through ndtrack With a Specific DBA

When you choose to collect Oracle inventory without installing FlexNet inventory agent on the target Oracle servers, you can manually deploy and run `ndtrack`. In its default operation, `ndtrack` runs as the local operating system user currently logged into the Oracle server. You can also configure the inventory beacon to collect Oracle inventory using a specific sysdba account. With this non-recommended method of collecting Oracle inventory, you have to use third-party tools to collect Oracle inventory by running `ndtrack`:

Note: *This feature is only available for UNIX-based Oracle servers. Flexera Software recommends using agent-based or zero touch inventory collection methods.*

- As `root` using the default sysdba account

- As `root` with specified sysdba account
- As operating system user that has sysdba privileges
- As externally-authenticated account
- Remotely from inventory beacon
  With all of the above methods to collect inventory, you can use the `UploadLocation=<URL>` option of `ndtrack` to set the location where you want the discovery (`.disco`) and inventory (`.ndi`) files to be uploaded. For more information on the `ndtrack` options, see the *FlexNet inventory Agent and Managed Devices* guide that is accessible from the title page of the online help.

Running `ndtrack` locally as `root` using the logged-in operating system user

When `ndtrack` runs as root on the Oracle server with no database user specified, it runs as the local operating system user currently logged into the Oracle server. Through this default way of working, the `ndtrack` collects Oracle Database and application inventory with complete software and hardware inventory. Use the following command to collect inventory:

```
ndtrack.sh -t machine
```

Running `ndtrack` locally as root using a specific sysdba account

When `ndtrack` runs as `root` using a specific database administrator account name that has the `sysdba` access to the database, it collects Oracle Database and application inventory with some (not complete) hardware and other software inventory. The specified user must be a member of the `dba` group in `/etc/group`. If you do not specify the user, the `ndtrack` uses the first user from the `dba` database group. Use the following command to collect inventory:

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

where the specified user has the `sysdba` access to the database or it is a member of the `dba` database group on the local OS.

Running `ndtrack` locally as local operating system user that is also a sysdba

When `ndtrack` runs as local operating system user on the Oracle server to collect Oracle inventory, it collects Oracle Database and application inventory with some (not complete) hardware and other software inventory. The logged-in local operating system user must be a member of the `dba` group. Use the following command to collect inventory:

```
ndtrack.exe -t machine
```

Running `ndtrack` locally as an externally-authenticated account

FlexNet Manager Suite also allows you to collect Oracle inventory by running `ndtrack` as an externally-authenticated user. An externally-authenticated user is maintained by Oracle Database, but authenticated by the operating system, and not by Oracle Database. Follow these steps:

1. Configure a schedule task (using third-party tools) to run `ndtrack` on the Oracle server.
2. Set up a special externally-authenticated 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). Flexera Software provides a script to create and configure this database user, at Script to create an Oracle user with the access required for Oracle Database inventory collection (you will need to log into the Flexera Software Knowledge Base).
3. Run the following command to collect inventory:

```
ndtrack.sh -t machine -o OracleInventoryAsSysdba=false -o
OracleInventoryUser=<externally-authenticated user>
```

Note: *You can also set the following parameters in the `config.ini` and use the `ndtrack.sh -t machine` command to collect inventory. The `config.ini` file is present in the `/var/opt/managesoft/etc` directory.*

- *`OracleInventoryAsSysdba=false`*
- *`OracleInventoryUser=<externally-authenticated user>`*

Running `ndtrack` remotely from the inventory beacon

You can also configure the appropriate inventory beacon to collect Oracle inventory using a specific sysdba account. Follow these steps:

1. 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`

2. Ensure that you have created a user with `ssh` privileges and registered its credentials in 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. The specified user must be a member of the `dba` database user group in `/etc/group`. For more information, see the topics under **Password Management Page** in the online help.
3. Add the following code in the `ndtrack.ini` file present either in `%ProgramFiles%\Flexera Software\Inventory Beacon\RemoteExecution\Public\Inventory` or `%ProgramFiles(x86)%\Flexera Software\Inventory Beacon\RemoteExecution\Public\Inventory` folder on the appropriate inventory beacon:
   - When using a database user that is a member of the `dba` group: `[Managesoft/Tracker/CurrentVersion] OracleInventoryUser=oracledba`

- When using an externally-authenticated database user: `ndtrack.sh -t machine -o OracleInventoryAsSysdba=false -o OracleInventoryUser=<externally-authenticated user>`

4. The inventory beacon now collects Oracle inventory according to the appropriate discovery and inventory rule schedule.

Appendix F- Oracle Standard Users Exempted From Consuming Licenses

FlexNet Manager Suite automatically exempts the following standard named Oracle users from consuming licenses on Oracle instances:

- ABM
- AD
- AD_MONITOR
- AHL
- AHM
- AK
- ALR
- AME
- AMF
- AMS
- AMV
- AMW
- AN
- ANONYMOUS
- AP
- APPLSYS
- APPLSYSPUB
- APPS
- APPS_MRC
- AR
- AS
- ASF
- ASG
- ASL
- ASN
- ASO
- ASP
- AST
- GCS
- GHR
- GL
- GMA
- GMD
- GME
- GMF
- GMI
- GML
- GMO
- GMP
- GMS
- GMW
- GNI
- GR
- HCA
- HCC
- HCN
- HCP
- HCT
- HR
- HRI
- HXC
- HXT
- IA
- IAM
- IBA
- IBC
- ORASSO_DS
- ORASSO_PA
- ORASSO_PS
- ORASSO_PUBLIC
- ORDPLUGINS
- ORDSYS
- OSE$HTTP$ADMIN
- OSM
- OTA
- OUC
- OUTLN
- OWA_MGR
- OWAPUB
- OZF
- OZP
- OZS
- PA
- PAY
- PBR
- PER
- PERFSTAT
- PFT
- PJI
- PJM
- PM
- PMI
- PN
- PO

- AU
- AURORAJISUTILITY$
- AURORAORBUNAUTHENTICATED
- AX
- AZ
- BEN
- BIC
- BIE
- BIL
- BIM
- BIN
- BIS
- BIV
- BIX
- BIY
- BLC
- BLEWIS
- BNE
- BOM
- BSC
- CCT
- CDOUGLAS
- CDR
- CE
- CHV
- CLA
- CLE
- CLJ
- IBE
- IBP
- IBT
- IBU
- IBW
- IBY
- ICX
- IEB
- IEC
- IEM
- IEO
- IEP
- IES
- IET
- IEU
- IEV
- IEX
- IGC
- IGF
- IGI
- IGS
- IGW
- IMC
- IMT
- INTERNET_APPSERVER_REGISTRY
- INV
- IP
- IPA
- POA
- POM
- PON
- PORTAL
- PORTAL_APP
- PORTAL_DEMO
- PORTAL_PUBLIC
- PORTAL30
- PORTAL30_DEMO
- PORTAL30_PUBLIC
- PORTAL30_SSO
- PORTAL30_SSO_PS
- PORTAL30_SSO_PUBLIC
- POS
- PQH
- PQP
- PRP
- PSA
- PSB
- PSP
- PSR
- PTE
- PTG
- PTX
- PV
- QA
- QOT
- QP

- CLKRT
- CLL
- CLN
- CN
- CPGC
- CRP
- CS
- CSC
- CSD
- CSE
- CSF
- CSI
- CSL
- CSM
- CSN
- CSP
- CSR
- CSS
- CST
- CTB
- CTSYS
- CUA
- CUC
- CUE
- CUF
- CUG
- CUI
- CUN
- IPD
- IPM
- IRC
- ISC
- ISX
- IT_PERF
- ITA
- ITG
- IZU
- JA
- JE
- JG
- JL
- JMF
- JTF
- JTI
- JTM
- JTR
- JTS
- JUNK_PS
- KWALKER
- LNS
- MDSYS
- ME
- MFG
- MIA
- MIV
- MQA
- QRM
- QS
- QS_ADM
- QS_CB
- QS_CBADM
- QS_CS
- QS_ES
- QS_OS
- QS_WS
- RCM
- REPADMIN
- RG
- RHX
- RLA
- RLM
- RMG
- RRC
- RRS
- SCOTT
- SH
- SHT
- SPIERSON
- SQLAP
- SQLGL
- SSOSDK
- SSP
- SYS
- SYSADMIN

- CUP
- CUR
- CUS
- CZ
- DBSNMP
- DCM
- DDD
- DEM01
- DISCOVERER5
- DNA
- DOM
- DSGATEWAY
- DSSYS
- DT
- DUMMY_GMO
- EAA
- EAM
- EC
- ECX
- EDR
- EGO
- EMS
- ENG
- ENI
- EVM
- FA
- FEM
- FF
- MRP
- MSC
- MSD
- MSO
- MSR
- MST
- MWA
- OAM
- OCA
- ODM
- ODM_MTR
- ODQ
- ODS
- ODSCOMMON
- OE
- OFA
- OKB
- OKC
- OKC_REP_TXT_INDEX_OPTIMIZE
- OKC_REP_TXT_INDEX_SYNC
- OKE
- OKI
- OKL
- OKO
- OKP
- OKR
- OKS
- OKT
- SYSTEM
- TEST
- TRACESVR
- UDDISYS
- VEA
- VEH
- WFADMIN
- WH
- WIP
- WIRELESS
- WK_PROXY
- WK_Test
- WKSYS
- WMA
- WMS
- WMSYS
- WPS
- WSH
- WSM
- XDB
- XDO
- XDP
- XLA
- XLE
- XNA
- XNB
- XNC
- XNI

- FII
- FLM
- FND
- FPA
- FPT
- FRM
- FTE
- FTP
- FUN
- FV
- OKX
- OLAPDBA
- OLAPSVR
- OLAPSYS
- ONT
- OPI
- ORAOCA_PUBLIC
- ORASSO
- XNM
- XNP
- XNS
- XNT
- XTR
- ZFA
- ZPB
- ZSA
- ZX

6

Server Scheduling

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.

For cloud-based implementations, this chapter provides only background reading for information. None of the low-level scheduling of the Microsoft Task Scheduler or the FlexNet batch scheduler and batch processor is accessible to those using the cloud implementation. Instead, tenants in the cloud systems can schedule their FlexNet inventory imports and license consumption calculations by navigating to the **system menu > SystemSettings > Inventory** tab, under **Managing the processing queue for imports and reconciliation**.

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.

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.

Scheduled task	Default schedule	Command	Note
FlexNet inventory data maintenance	Midnight daily	`BatchProcessTask.exe run IMDataMaintenance`	Performs cleanup on the inventory database (containing FlexNet inventory) in FlexNet Manager 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`	Reserved for future development.
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.

Scheduled task	Default schedule	Command	Note
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).
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).

Scheduled task	Default schedule	Command	Note
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 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.

Scheduled task	Default schedule	Command	Note
Import SAP package license	6am Sunday	`BatchProcessTask.exe run SAPPackageLicenseImport`	Imports the licenses calculated by SAP, writing results into the compliance database.
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).
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.

Scheduled task	Default schedule	Command	Note
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.
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

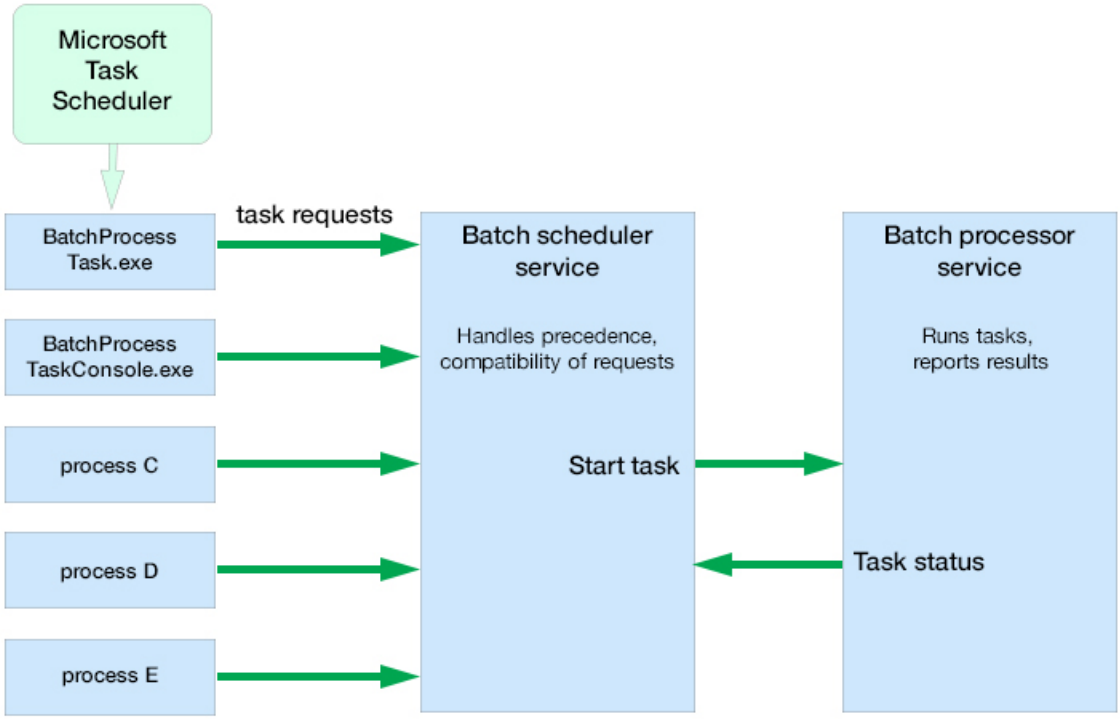
The batch scheduler for FlexNet Manager Suite runs on the batch server. It is responsible for receiving messages that request specific tasks, and placing these in an execution queue in such a way as to eliminate incompatibilities and (to some degree) optimize performance.

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.)
   - The scoping is correctly supplied, as listed in Batch Scheduler Command Line (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.
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.
`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).

Status	Updated by	Description
`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 inventory beacons. The purpose of these specializations is to make third-party inventory results available in the web interface as promptly as possible.

Third-party inventory processing overview:

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.)
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.

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.

Synopsis

Syntax:
`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:

Syntax:
`-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)

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 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.

Task	Scope	Notes
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`

Task	Scope	Notes
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.* 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.*

Task	Scope	Notes
`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 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.

Task	Scope	Notes
`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.* 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`

Task	Scope	Notes
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.
IBMPassportAdvantage	Tenant	Used for import of IBM Passport Advantage reports through the web interface. Not available for external use. Underlying executable: SQL stored procedure.

Task	Scope	Notes

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.

Task	Scope	Notes
`IMImport`	Tenant	Runs the inventory import for FlexNet inventory (collected either by the FlexNet inventory agent locally installed on the inventory device, or by the FlexNet inventory core components operating from an inventory beacon to take zero-footprint inventory). 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).
		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`

Task	Scope	Notes
LicenseReconcile	Tenant	Runs only the license consumption calculations of the compliance writers (that is, does not include writing the 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. The additional verbs covered here are generally reserved for development, testing, and remediation.

Syntax:
`BatchProcessTaskConsole.exe help`

Prints a text message to the console showing the command-line options.

Syntax:
`BatchProcessTaskConsole.exe list-tasks`

Creates a list of tasks that are currently executing, together with requests for tasks to execute in future.

Syntax:
`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.

Syntax:
`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.

Syntax:
`BatchProcessTaskConsole.exe test` *taskName* [*options*] [**`--`** *taskArguments*]

Use with the same task names and arguments as listed in Batch Scheduler Command Line. 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.

Syntax:
`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.

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`
- `IMImport`
-
- `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). 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"
```

7

The Inventory Adapter Studio

Inventory Adapter Studio is a tool to develop adapters for custom inventory gathering into FlexNet Manager Suite.

This section covers the use of Inventory Adapter Studio and the management of the adapters you develop.

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 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 must be 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 the central server in the cloud.
- **File access.** On the beacon 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.** Inventory data gathered by the adaptors created on the inventory beacon is uploaded automatically to the hosted service, and imported into the operations database there.

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 theinventory beacon for adapter development and testing. This may be on the beacon 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.

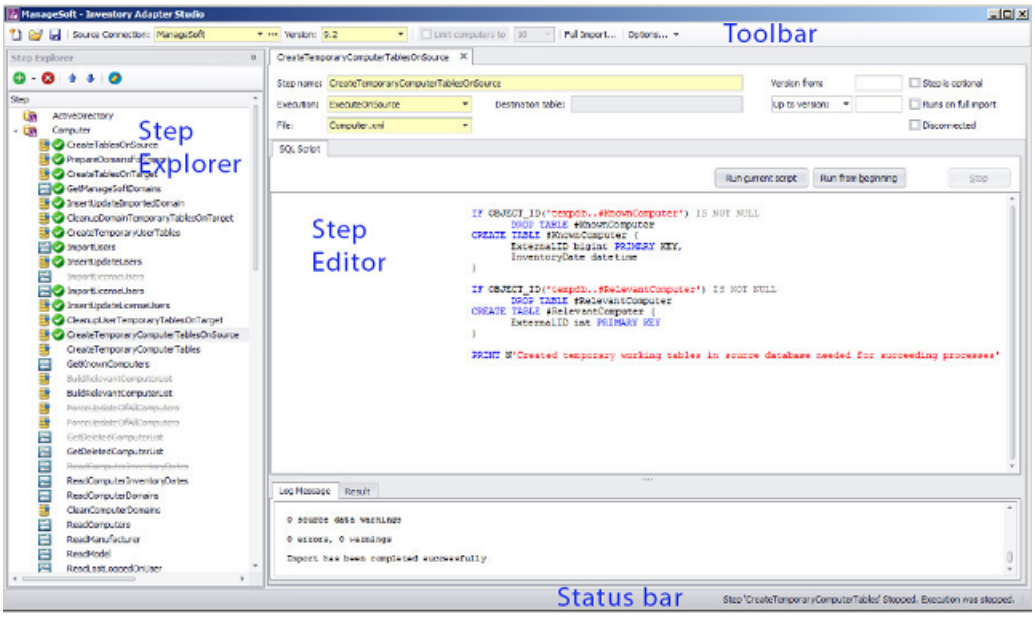
Restrictions

Several inventory adapters are supplied as standard with FlexNet Manager Suite (these are sometimes referred to as "Tier 1" adapters). These live on the central server, and are automatically downloaded to inventory beacons as required. This means:

- You cannot modify Tier 1 inventory adapters.
- 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 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:



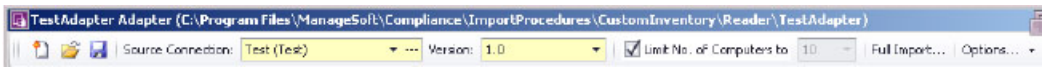
Element	Purpose
Toolbar	• Creates new adapters or open existing ones. • Manages database connections. • Saves changes. • Specifies the database connection to work on. • Specifies the data size limits to apply when testing.

Element	Purpose
Step Explorer	Shows the steps in the currently open adapter. These open the edit panels on the right. • Import execution status will show as icons in this element. • Bold steps in the templates show where user customization is required; other steps have been completed by Flexera Software. • Steps may be added, deleted or have their execution order changed using the toolbar in the step explorer.
Step editor	Shows: • The name of the step and its settings. • The script in the step, with a Run button for testing. • The logs, showing import results. • The Result panel, which shows datasets from your SQL queries.
Status bar	Shows import progress.

Each section is discussed in more detail in the following topics.

Toolbar

The toolbar contains the following controls:



New button

A button that launches the **Create New Adapter** dialog.

Open button

A button that launches the **Open Existing Adapter** dialog. This allows browsing of all custom and factory-supplied adapters.

Save button

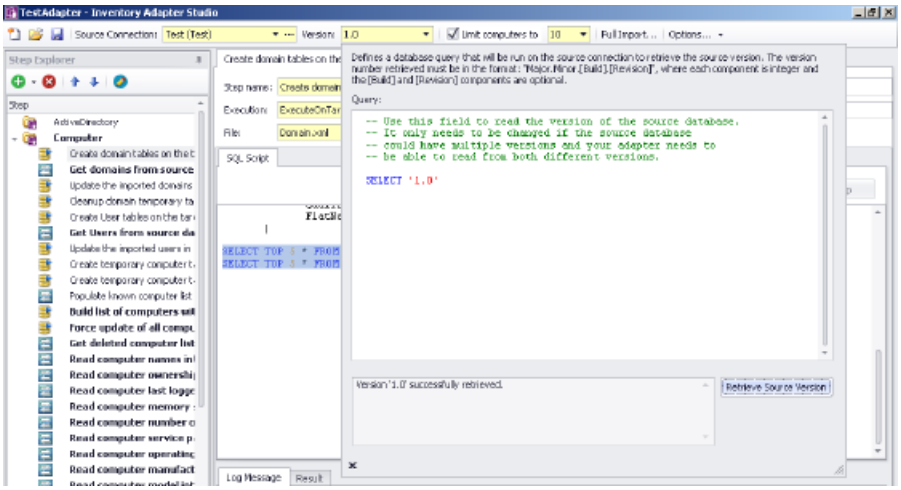
The Save button saves all files in the adapter that is being edited. This includes changes due to steps being moved in the Step Explorer, or versions being changed in the Version field on the toolbar.

Source Connection control

The drop-down portion of this control allows selection of an existing database connection. New connections can be created and existing connections may be edited using the '...' button, which launches the **Select Source Connection** dialog.

Version control

This control shows the version of the currently selected source connection. This version is evaluated by executing a query against the source database. Clicking the drop-down button displays a dialog that allows you to change this query for an adapter. Clicking the **Retrieve Source Version** button will execute the query and show the results in the dialog.



Use of this control is particularly important if your adapter supports importing inventory data from multiple versions of the same system. This usually occurs as enterprises upgrade systems over time.

Limit Number of Computers control

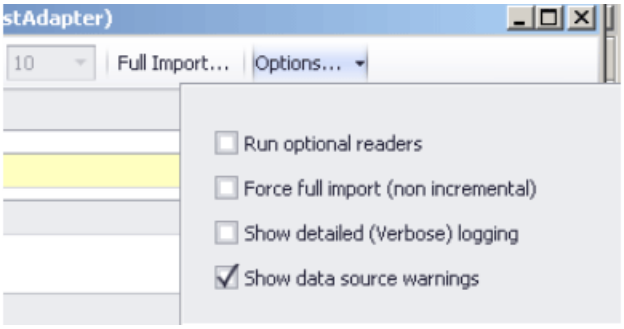
This control is checked and enabled by default for test connections. When set to a value, it limits the number of computers read by an adapter.

Full Import button

This button launches the **Full Import** dialog, which allows you to execute your adapter end to end and check your results in FlexNet Manager Suite.

Options

There are several options that apply to the Run buttons on the Edit panel. These control the way the Compliance Importer executes your adapter and correspond to command line arguments.



Options include:

- **Run optional readers**: the next run command will exercise steps marked with the **Step is optional** attribute.
- **Force full import**: the next run command includes steps marked with the **Runs on full import** attribute.

Tip: *When the attribute values prevent the execution of the step in the next run, it is greyed out with a strike-through in the step explorer.*

Step Explorer

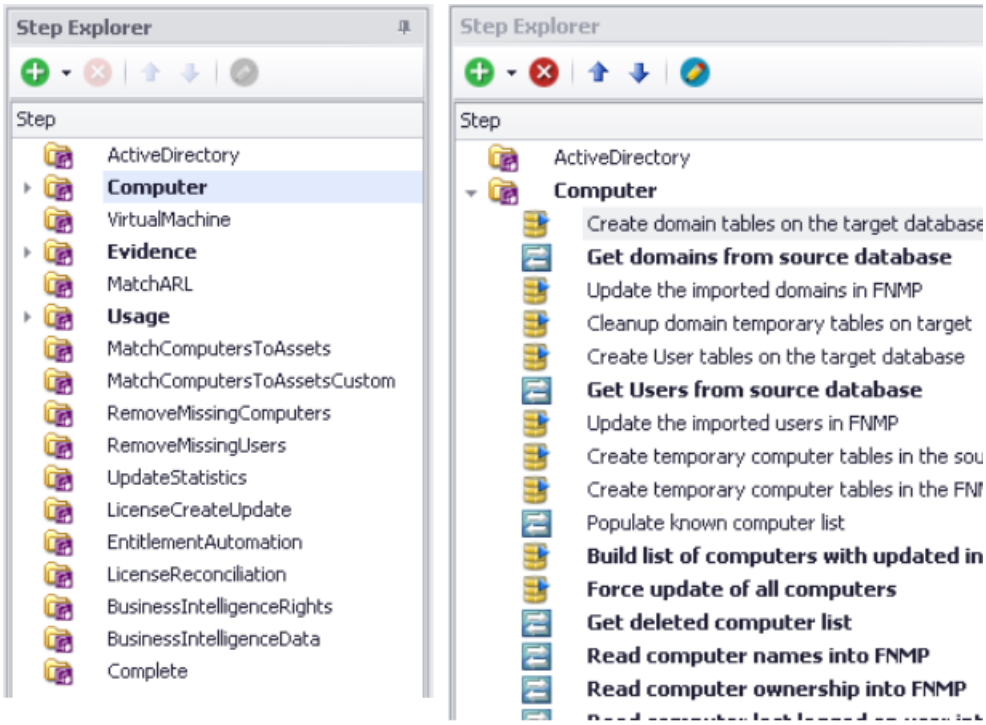
The step explorer shows the procedures in the Compliance Importer, and the steps within those procedures that will be executed for the current adapter.

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.

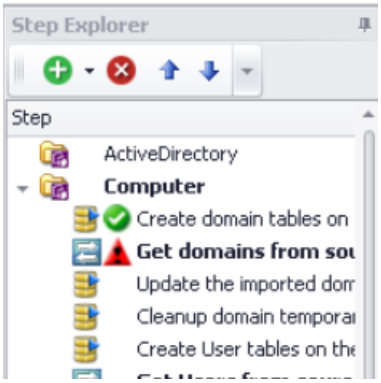
Figure 3: Collapsed (left) and expanded, with bold steps requiring customization. Custom SQL and data transfer steps visible.



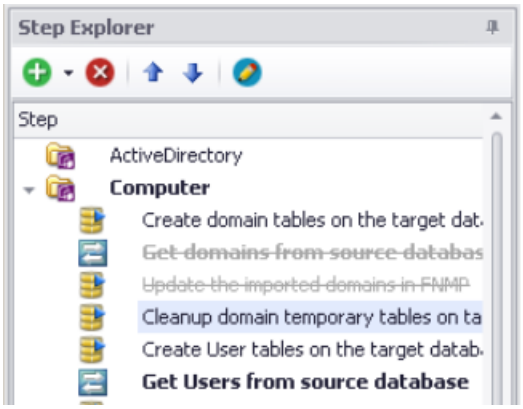
The toolbar allows steps to be added, removed, edited and to change their execution order. There are two types of steps that may be added:

- 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.



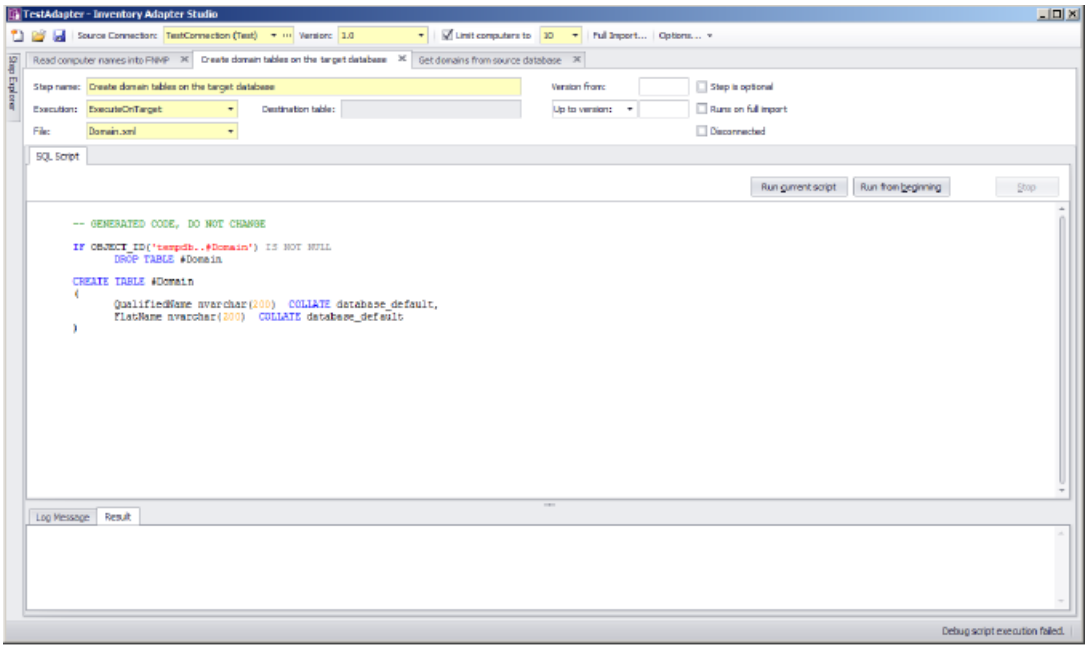
When the attribute values of a step will prevent it being executed for the version of the current connection, it will be greyed out with a strike-through in the step explorer.



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

In *disconnected mode* (when the inventory adapter is running on your inventory beacon), some values are available for your custom inventory adapters, while others may appear only in factory-supplied adapters, as shown in the table below. DO NOT use these values in your custom adapters, as these steps are blocked (for security reasons) in disconnected mode, and using them will cause your inventory import to fail.

Value	Step type	Notes
`ExecuteOnSource`	Custom SQL	The SQL script will run on the source database.
`ExecuteOnTarget`	Custom SQL	Factory use only. DO NOT modify. DO NOT use in custom inventory adapters.
`GetVariableFromSource`	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`	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`	Data Transfer	Factory use only. DO NOT modify. DO NOT use in custom inventory adapters.

Value	Step type	Notes
`TargetToSource`	Data Transfer	Factory use only. DO NOT modify. DO NOT use in custom inventory adapters.

Destination table

Do not use for inventory adapters running on your inventory beacon (that is, in disconnected mode). Factory-supplied standard adapters may display values in this field. Do not alter such an adapter.

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.

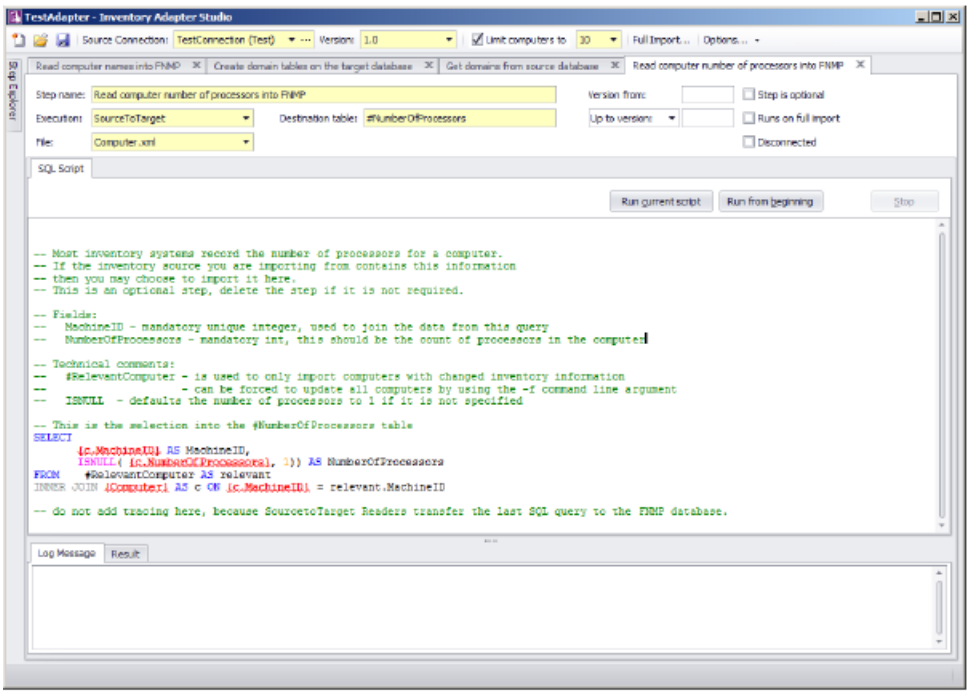
Stop

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 4: Red underlined text should be replaced with your own database column and table names.



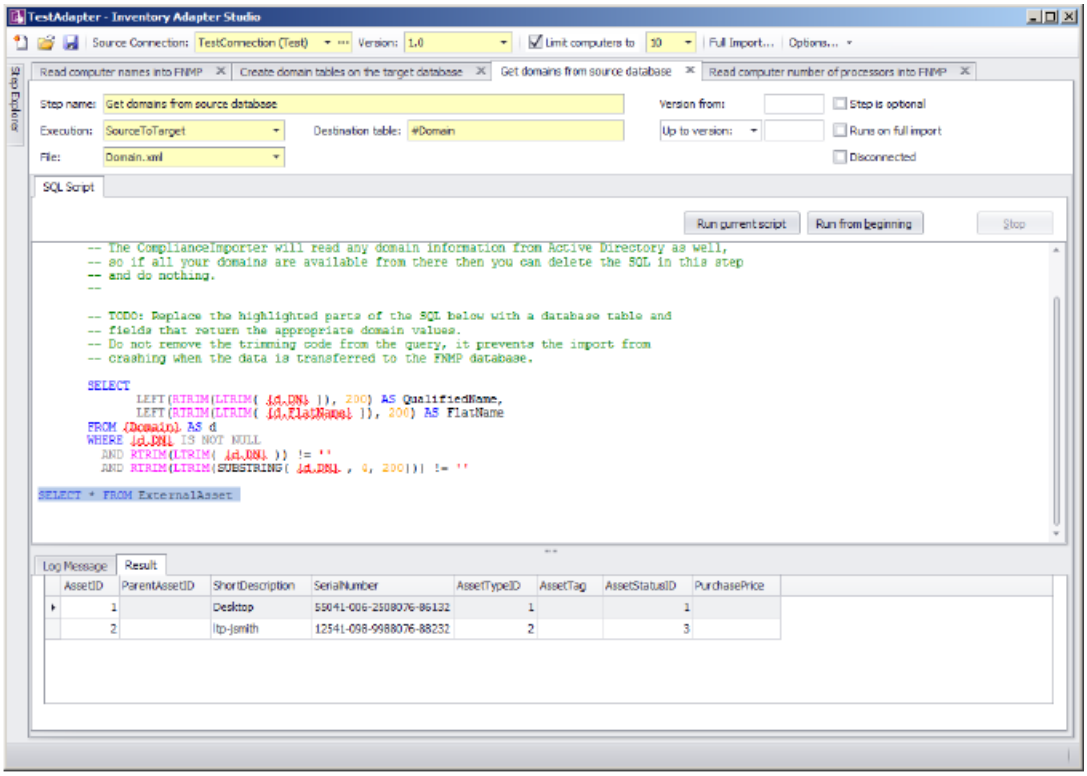
Log Message and Result panel

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

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`
Template file storage: `C:\ProgramData\Flexera Software\Compliance\ImportProcedures\`, followed by:

- `AdapterStudioTemplates` for templates downloaded from the central application server
- `CustomInventory` for use in an on-premise installation (not used for FlexNet Manager Suite as a cloud service)
- `Inventory` for standard adapters supplied with the product. These implement standard integration with other products. You must not edit these in any way, or they will cease to operate and cause all attempts to import using them to fail completely.
- `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).

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 provided for intermediate storage and workspace for data used by each procedure.
- 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. When operating in disconnected mode on an inventory beacon (as is always the case with FlexNet Manager Suite as a cloud service), 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 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.

Understanding the function of the Reader procedures is especially important to preparing inventory adapters.

Structure of an Inventory Adapter

Each inventory adapter is a set of reader instructions for the compliance importer. The permitted structure depends on the origin and operational mode for this adapter.

Each adapter runs in one of (up to) three modes:

- "Connected mode", where the adapter has simultaneous access to both the source and target databases (for example, when the source database is accessible from the server running the operations database for FlexNet Manager Suite).

  Note: *For security reasons, connected mode is not available when you are using FlexNet Manager Suite as a cloud service.*

- "Disconnected mode", where you need to install an inventory beacon, either because the source database and target database are on separate networks, or because you are using FlexNet Manager Suite as a cloud service. For more information see Disconnected Mode.
- "Disconnected mode (Tier 1)", where the same operational conditions apply with the adapter running on an inventory beacon, but because the adapter is factory-supplied, security provisions take a different form, discussed below.

The operations that are available when creating a custom adapter depend on the mode in which it runs.

Note: *Adapters engineered by Flexera 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.*

Table 7: Availability of all adapter operation types

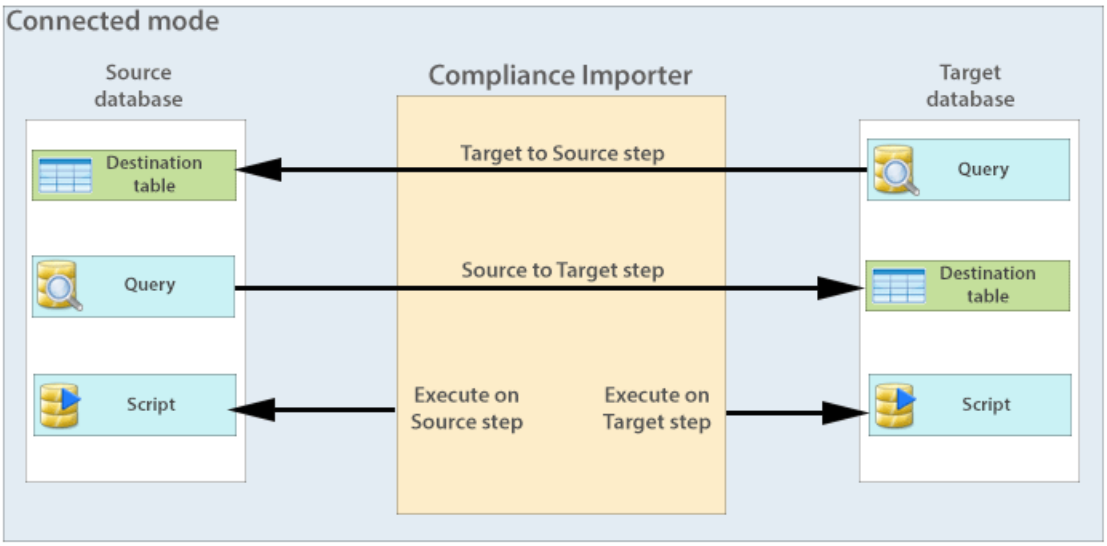
Type of operation	Description	Available in modes
Target to source	Executes a database query on the FlexNet Manager Suite database and copies the result to a table in the source database. This data is to provide context for a subsequent query on the source database.	Connected

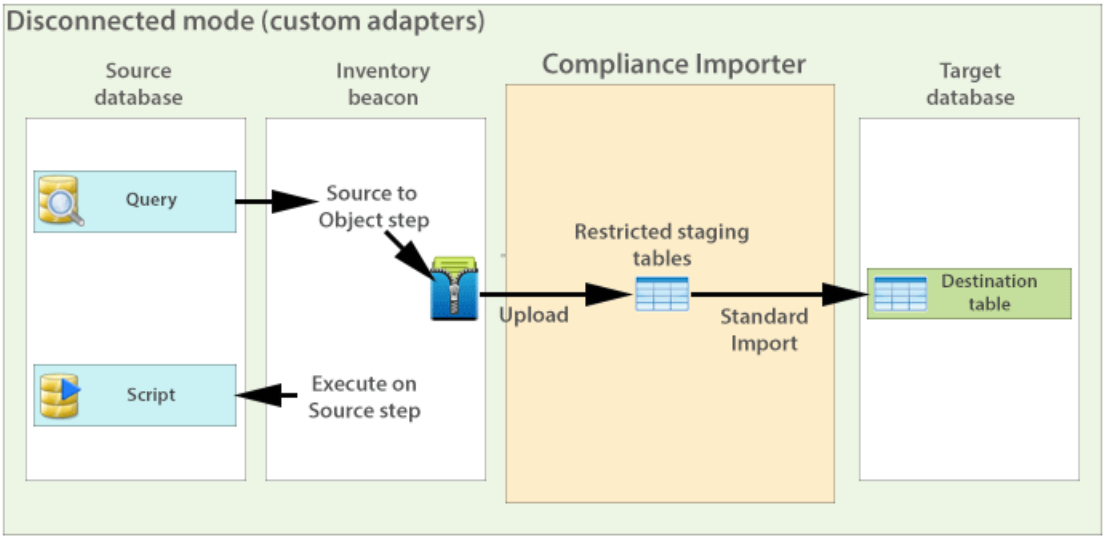
Type of operation	Description	Available in modes
Source to target	Executes a database query on the source database and copies the result to a table in the FlexNet Manager Suite database. ***Restriction:*** *For fast transfers, the Compliance Importer uses SQL bulk copy operations to move data from one database to another. This means that the query on the source and the table on the target must match exactly in column order and data type.* For disconnected (tier 1) operations, a source to target step is modified so that data is saved to intermediate packages before upload.	Connected Disconnected (Tier 1)
Source to object	Replaces 'source to target' for use in disconnected mode with custom adapters. Instead of writing source data directly to the target database, it is written into an intermediate package format and saved on the inventory beacon. The intermediate packages are uploaded asynchronously to the cloud server, and processed (by default overnight) for import into the operations database. ***Restriction:*** *Since you cannot modify the internal processing, the data in the intermediate packages must map correctly to a standard set of objects and their attributes (see Inventory Adapter Object Model).*	Disconnected
Execute on source	Executes an SQL script on the source database.	Connected Disconnected (switchable for disconnected *only*) Disconnected (Tier 1)
Execute on target	Executes an SQL script on the FlexNet Manager Suite database.	Connected Disconnected (Tier 1)

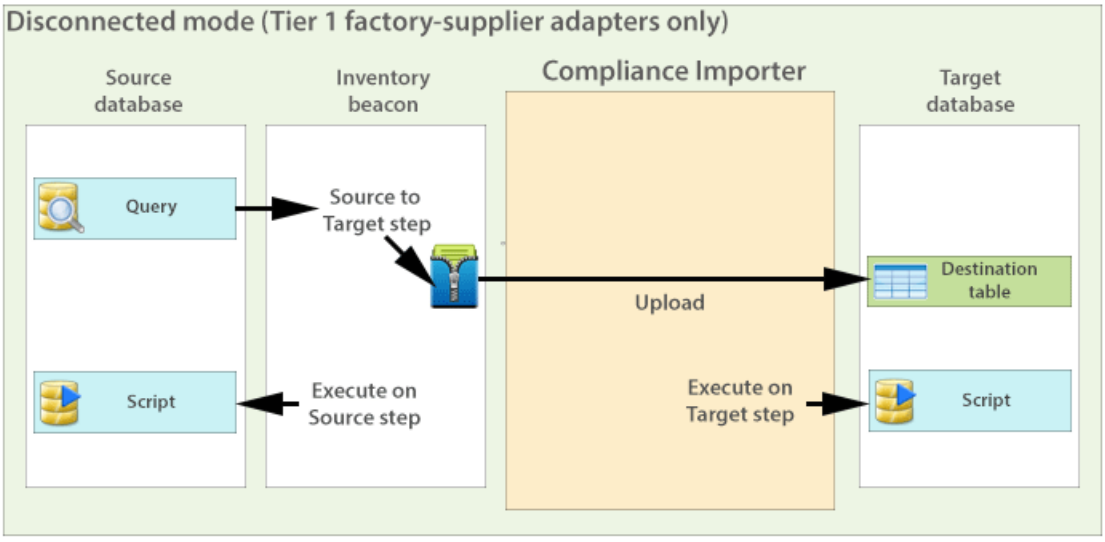
The three architectures are shown in the diagrams below. From a security perspective, the distinction between the modes is:

- Connected mode applies only for on-premises installations where the security of both databases is entirely in your control.
- Disconnected mode for custom adapters protects the central operations database in the cloud service by disallowing any custom SQL on the target side. This also limits the permissible imports to the standard set of objects and attributes available in the Inventory Adapter Studio running on an inventory beacon. You can also find an XML file defining those objects and attributes on your inventory beacon at `C:\ProgramData\Flexera Software\Compliance\ImportProcedures\ObjectAdapters\InventoryObjectModel.xml`.

- In disconnected mode, Tier 1 (factory-supplied) adapters are able to use factory-approved custom SQL on the target side. Security is provided by disallowing the slightest revision of any kind to these adapters. If you change anything on Tier 1 inventory adapters for disconnected mode, they will not run, and importing your inventory will completely fail.


Connected mode
Source database
Compliance Importer
Target database
Destination table
Target to Source step
Query
Query
Source to Target step
Destination table
Script
Execute on Source step
Execute on Target step
Script


Disconnected mode (custom adapters)
Source database
Inventory beacon
Compliance Importer
Target database
Query
Source to Object step
Restricted staging tables
Destination table
Upload
Standard Import
Script
Execute on Source step


Disconnected mode (Tier 1 factory-supplier adapters only)
Source database
Inventory beacon
Compliance Importer
Target database
Query
Source to Target step
Destination table
Upload
Script
Execute on Source step
Execute on Target step
Script

Structure of Templates for Inventory Adapters

Templates are provided for inventory adapters, to speed your development effort.

The adapter templates shipped with the Inventory Adapter Studio use the adapter structure as follows:

- Temporary database tables are set up on the source and FlexNet Manager Suite databases.
- Sample queries are written in Source to Target steps. These write to the temporary tables already created.
- To make each query as easy to write as possible, the minimum number of columns is sent in each query. This also documents the minimum requirements for importing the inventory source.
- Areas of the query that require change are enclosed with curly brackets, colored red, and underlined: {Replace this text}
- As many of the other steps as possible are already completed. They rely on the fact that data has been transferred in the Source to Target steps.

Each step in the templates has comments describing the updates that need to occur. Optional fields are identified, and the adapter will still work if the optional fields are not provided.

At the end of each procedure there is a lot of provided code that performs a differential update into the Imported database tables in the FlexNet Manager Suite database. It is recommended that you do not change this code for your adapter. There may be special cases where this is required, but an error will prevent any data from being imported into FlexNet Manager Suite.

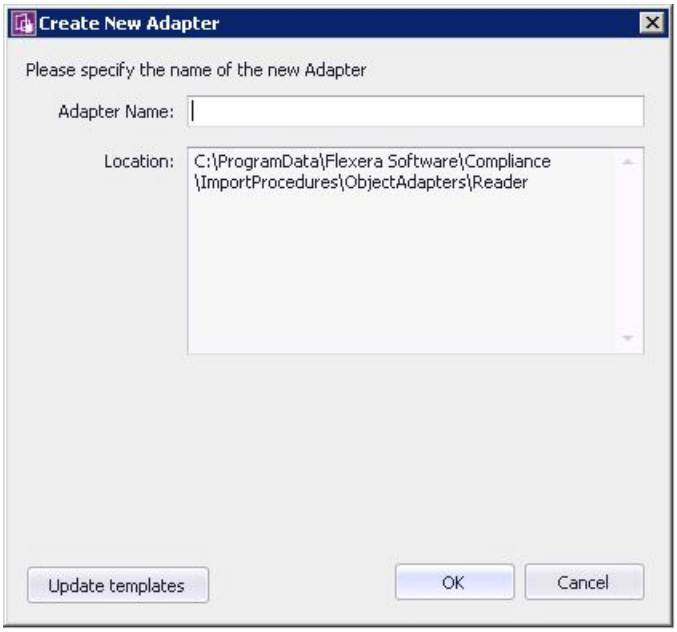
To Create a New Adapter

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.

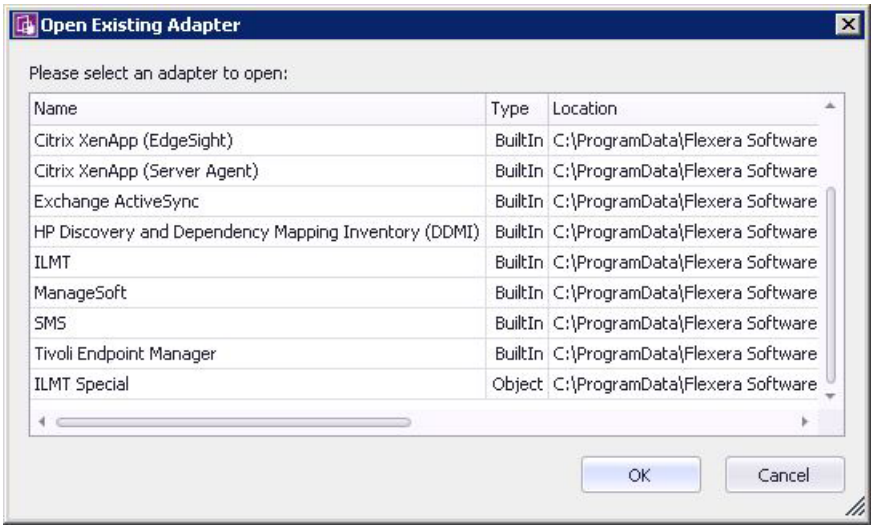
You may edit your custom adapters. In disconnected mode (that is, using the cloud service solution), do not attempt to edit any factory-supplied (Tier 1) adapters. If you edit any part of a Tier 1 adapter, it ceases to operate.

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.
- Adapters of type `Object` are those which you have previously edited.

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).

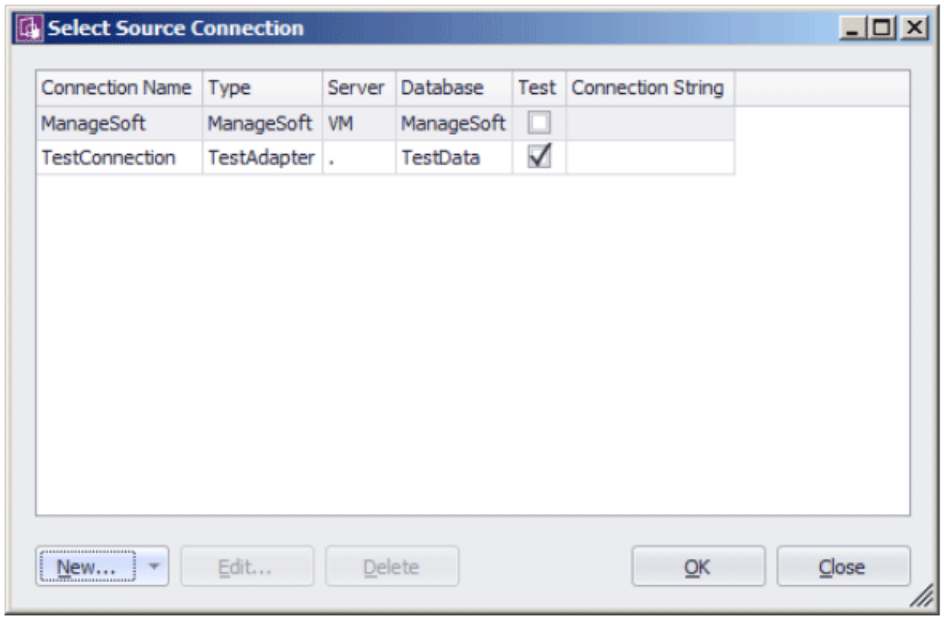
You must know either the server name or its IP address (together with database instance name, if any) to type in during the following process. (There is no browse facility to find the server.)

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.

Note: *Creating a new test connection adds this connection to the list available in the inventory beacon interface.*

1. In the toolbar, on right-hand end of the **Source Connection** control, click the [**...**] ellipsis button.

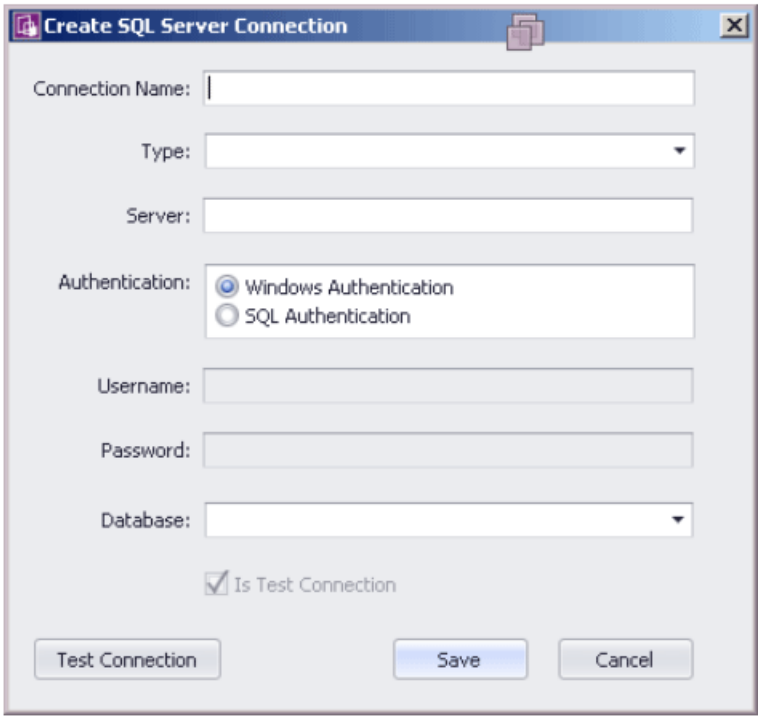
The **Select Source Connection** dialog opens.



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.

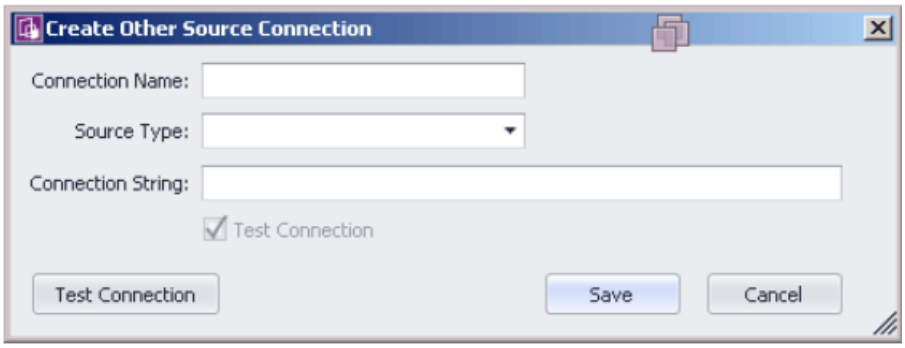
   f. In the **Database** field, type the name of the database, or use the pull-down list to select from database names automatically detected on your specified server.

   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).
2. Specify a new connection to a data source. Initially this is a test connection during your development phase. (See To Create a Source Connection.)
3. Use the **Step Explorer** to:
   - Add a new step to one of the grouping folders (see Adding a New Step to an Inventory Adapter)
   - Remove any steps provided automatically that are not required in your inventory adapter (see Remove a Step from an Inventory Adapter)
   - Change the execution order of steps within your inventory adapter (see Reorder Steps in an Inventory Adapter)
   - Update the details included within an individual step.

   Note: *You cannot add, delete, or reorder folders in the* ***Step Explorer****. These are optimized for the order of insertion into the operations database. You may only modify or reorder the steps within a given folder.*

4. Test each step in your adapter as you develop it (see Tips for Editing an Adapter and Testing an Adapter). Cycle through until you have created and tested all the steps needed to complete your inventory adapter.
5. Run the entire adapter, and validate that the collected data is as you expect (again, see Tips for Editing an Adapter). On an inventory beacon, validation means unzipping the intermediate package and examining the XML file it contains. Look for your created intermediate package in `C:\Program Data\Flexera Software\Beacon\IntermediateData`.
6. Put the completed and test inventory adapter into production (see To Publish Your Adapter).

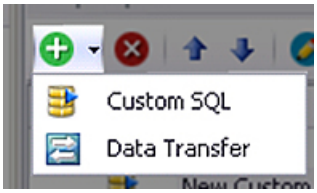
Adding a New Step to an Inventory Adapter

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.)

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 the source 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.

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.

Using FlexNet Manager Suite as a cloud service (software as a service, or SaaS), you have access only to your inventory beacon(s), with no direct access to the central operations database. This means your custom inventory adapters always run in *disconnected mode*.

However, factory-supplied inventory adapters may have some steps that are designed for on-premises implementations that may access a central operations database. Such steps, designed for connected mode, cannot run in disconnected mode (and are automatically skipped when the inventory adapter runs on an inventory beacon). For that reason, factory-supplied adapters may have alternate steps that run *only* when the inventory adapter runs on your inventory beacon, in disconnected mode. Such steps are differentiated by having the **Disconnected** check box set.

Tips for Editing an Adapter

The following principles are helpful when you are developing and testing your inventory adapter. If you need information about the fields in the editing pane and the permitted values, see Edit panel.

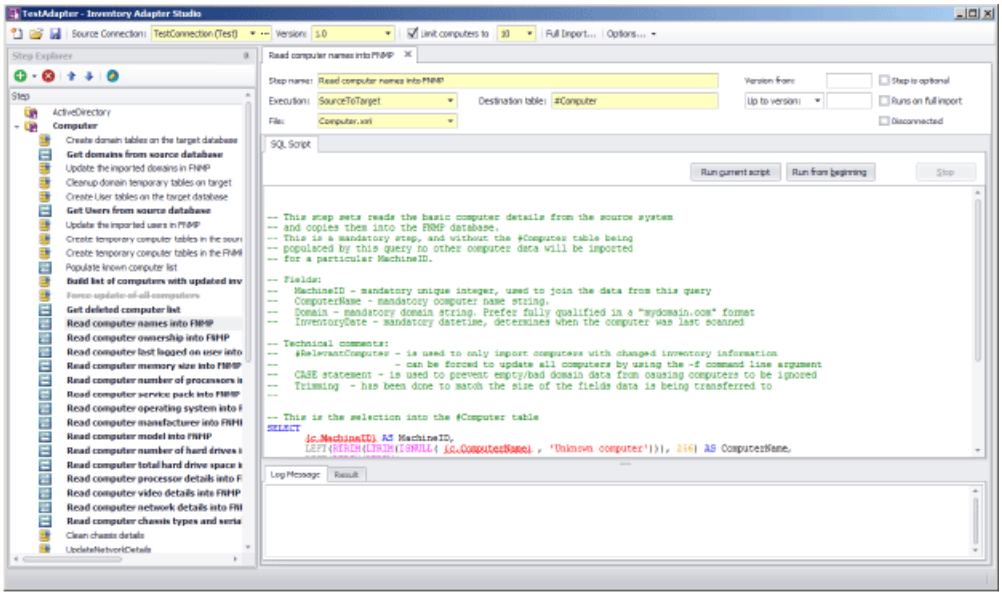
Minimize processing times

During development, use the setting to limit the number of computers for test imports. This will potentially save hours of processing while testing your adapter. The control applies a filter to the number of computers that the adapter reads from the source database, allowing validation of your work without requiring that all the data is read and processed.

Start with a template

Start with one of the supplied templates and customize it to suit your data source. Focus on the areas needing change:

- In the Step Explorer, each step that requires editing is shown in bold text. The bold is automatically removed when all areas requiring change have been modified.
- When you have selected a step so that its details are shown in the edit panel, the individual edits required are shown within curly braces, underlined, and in red.
- Every step that needs editing has extensive comments on the data structures and requirements provided in the step details. Following these guidelines will provide the quickest path to a working adapter.



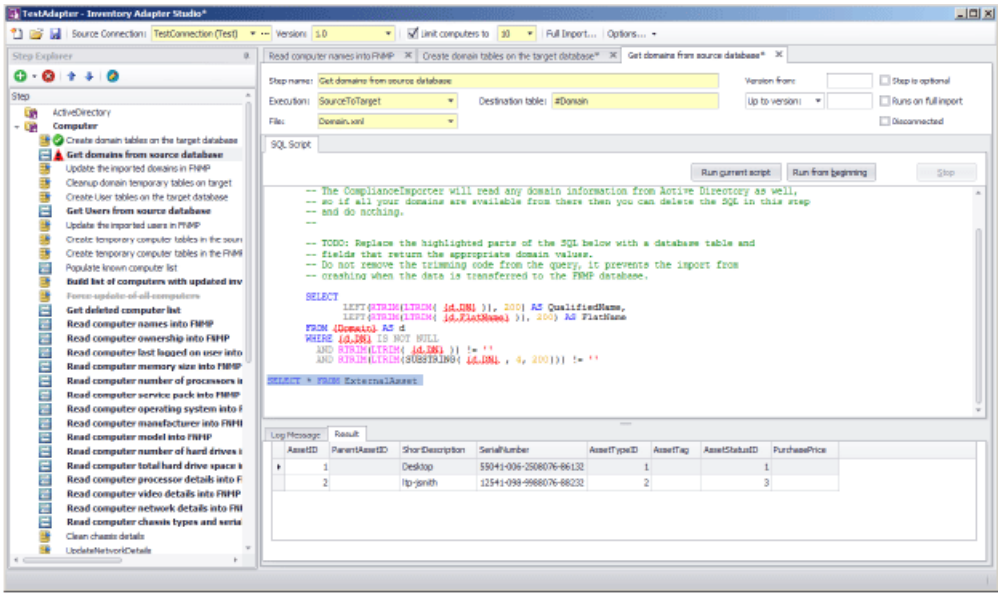
Test each SQL step

As you modify each step, you can test the SQL and inspect the data set it produces, by highlighting the section of your SQL to test, and clicking the **Run current script** button. The result is shown in the **Result** tab below.

Tip: *The **Run current script** button will run the entire step if there is no selection in the SQL script.*

Note: *If the selection (or, when there is no selection, the step) includes any red, underlined text that still requires customization, this produces a syntax error when run.*

Figure 5: 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.

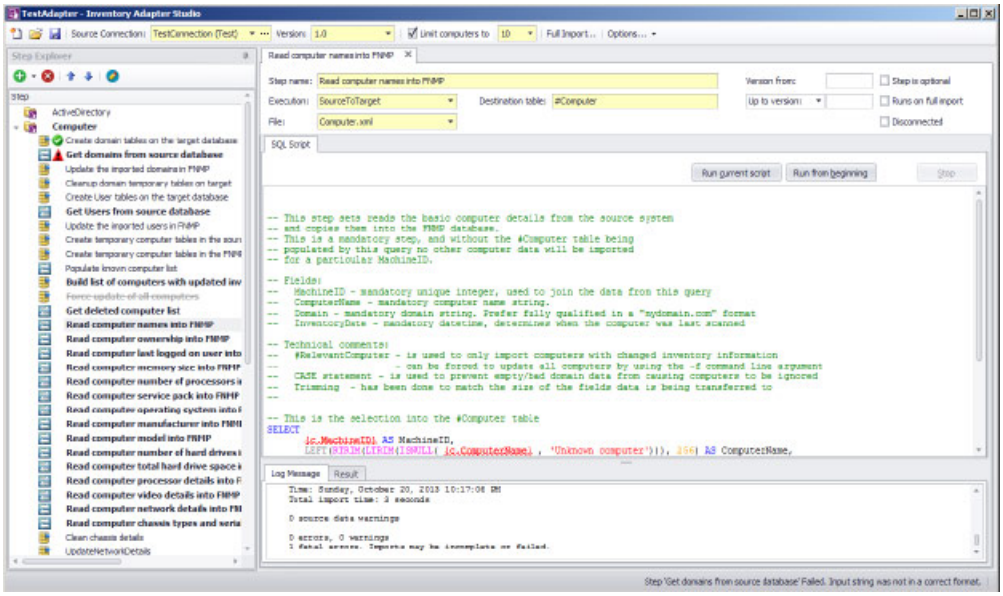


Test progressively

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.*

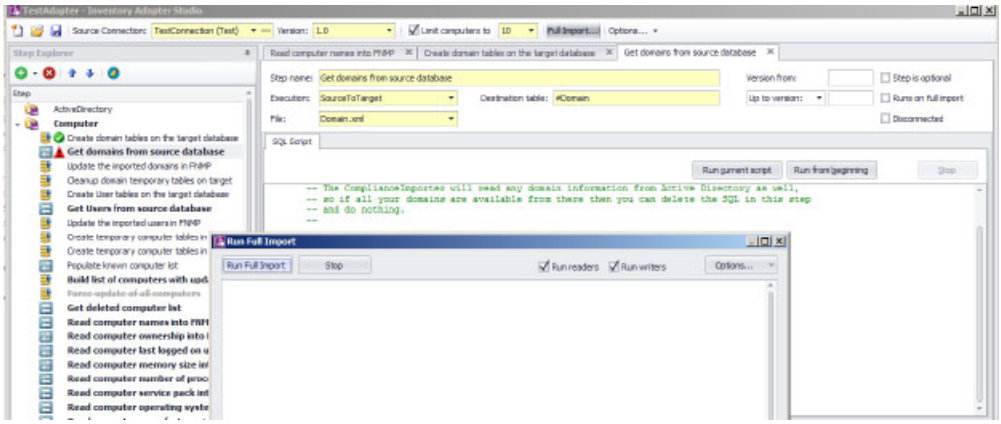


Final test

Finally, when you are ready to run an end-to-end test of your adapter, use the **Full Import** toolbar button.

Whereas your earlier testing displayed results in the Inventory Adapter Studio, a full import also writes the gathered inventory into a zipped archive ready for upload to the operations database. Look for your created intermediate package in `C:\Program Data\Flexera Software\Beacon\IntermediateData`. You can unzip the archive package, and examine the XML file it contains to ensure it contains all the required data.

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.



To Save an Adapter

Nothing unusual about this!

Save early, save often.

1. Your adapter is saved every time you do any one of the following:
    a. Click the Save icon in the toolbar.
    b. Use the `Ctrl-S` keyboard shortcut.
    c. Click the **Run from beginning** button to test the adapter.

    All the files are saved first, because running the adapter uses the Compliance Importer to read the files from disk. Anything not saved is not tested by the **Run from beginning** button.

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 and displaying results in the **Results** tab of the Inventory Adapter Studio.
- To validate the second stage, migrating the data from the staging tables into the operations database so that the data is visible in the compliance console,uploading the collected data to the FlexNet Manager Suite cloud database to that it becomes visible in the compliance browser, 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. 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 in a zip archive that packages version information, a package manifest, and the inventory data in a zip file awaiting upload to the operations database.

To Diagnose Readers for Your Adapter

Because the inventory adapter is running on your inventory beacon, you can only inspect the uploadable package to validate reader operations.

Follow this procedure using a plain text or XML editor of your choice on the inventory beacon.

1. Locate the uploadable package that the adapter has saved on the inventory beacon.

   By default, this is located in `C:\ProgramData\Flexera Software\Beacon\IntermediateData` (if this location is not in use, check the registry key `HKLM\SOFTWARE\ManageSoft Corp\ManageSoft\Beacon\CurrentVersion\BaseDirectory`, and append `IntermediateData` to the value found there). In test mode, your adapter creates a folder here named for your connection name. (Once this adapter is in production, this folder is replaced by a zipped archive named with the connection name plus 14 digits representing the date/time when the file was zipped.)

2. Examine the files in the folder. Each contains:

   - A version identifier - you cannot alter this by modifying your adapter (and should never edit this file in production).
   - A package manifest `packge.xml` - likewise, not configurable through your adapter, and do not edit in production.
   - An XML file containing your inventory information.

3. Review the contents of this XML inventory file to validate that the dataset is as expected.

   Within this file the `meta` element defines the destination database object, and the attributes represented in each logical column of data. Thereafter, each row contains the data in the matching columns.

To Diagnose Writers for Your Adapter

With the software as a service (SaaS) solution, you can inspect results in the compliance browser after the next upload and compliance calculation.

The simplest way to validate that your data is being imported all the way into the operations database is to inspect the results in the web interface.

1. To ensure the archived package is uploaded to the operations server, publish your adapter (by turning off the **Test** setting), and run a **Full Import**. Wait until the resulting upload is completed.

   Each full import triggers an upload, and there is a 'catch-up' scheduled upload to retry any failures scheduled overnight.

2. To ensure that the uploaded data is processed from the staging tables into the operations database, do either of the following:

- Wait until the next scheduled inventory import. By default, the inventory import and recalculation is triggered overnight.
- Trigger an inventory import/recalculation now. Be aware that this processes all current data, and is not restricted to your new inventory import. As a result, it may take some time (hours, for a large computer estate). Use the following steps:

  a. In your compliance browser, navigate to the **Reconcile** page (**License Compliance** > **Reconcile**).

  Restriction: *You must be in a role with administrator rights to be able to include your new inventory import in the reconciliation you run manually.*

  b. Select the **Update inventory for reconciliation** check box.

  This setting ensures that uploaded content is incorporated into the operations database and used for the compliance recalculation. Wait for the import and reconciliation to succeed (monitor the **Last successful reconcile** display on the right of the title bar, refreshing your browser page as necessary).

3. When the reconciliation is complete, examine your imported information, for example in the following locations:
   - The **Discovery & Inventory > All Inventory** page (in the `Inventory` group) shows you all the computers that are being imported, as well as the hardware properties that you have set (remember to check the column chooser). Consider filtering by **Created** date to isolate your new imports.
   - The **Enterprise > All Users** page shows all the users you have imported and the attributes that have been set.
   - The **License Compliance > All Evidence** page has separate tabs to show the installation evidence, file evidence, and access evidence (for application virtualization) that was imported.
   - The **License Compliance** > **All Applications** page shows all the application installations identified as a result of the evidence import. If your inventory revealed an application for the first time, check for a status of `Unmanaged`.

   Note: *If imported evidence did not match any existing application rules, the application does not show in any application list. It will appear only when the Application Recognition Library is updated with new rules incorporating your new evidence.*

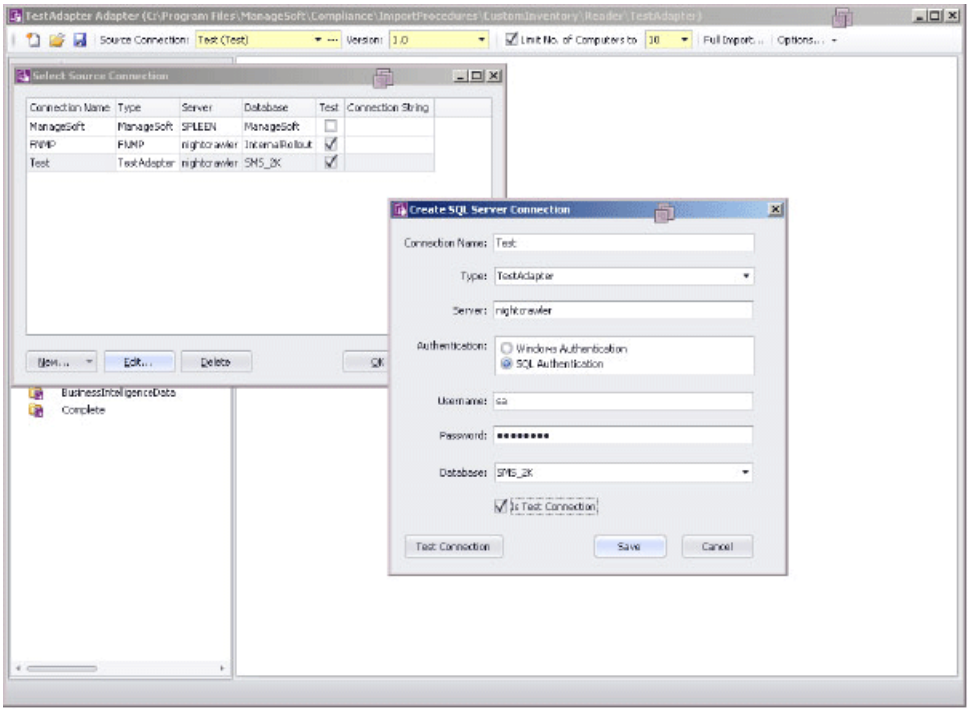
To Publish Your Adapter

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 inventory beacon. 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.

Here is a complete list of the database objects (and their permissible attributes) that you may import through a custom inventory adapter that runs on your inventory beacon.

Inventory Object: AccessingDevice

`AccessingDevice` objects are uploaded to the `ImportedAccessingDevice` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedAccessingDevice` table holds a record client access device information.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccessingDeviceID`	A numeric reference into a static table. May be null.	Matching accessing device ID. Foreign key to the `AccessingDevice` table.
`ComputerName`	Alpha-numeric text (maximum 256 characters). May be null.	Computer name of the client accessing device.
`Domain`	Alpha-numeric text (maximum 100 characters). May be null.	Domain name of the client accessing device.
`ExternalAccessing DeviceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used to identify the device in source connection
`IPAddress`	An ASCII string of alphanumeric characters and punctuation (length 256 characters). May be null.	IP Address of the client accessing device.
`SerialNo`	Alpha-numeric text (maximum 100 characters). May be null.	Serial no of the client accessing device.

Inventory Object: AccessingUser

`AccessingUser` objects are uploaded to the `ImportedAccessingUser` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedAccessingUser` table holds a record of the user access infomarion.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccessingUserID`	A numeric reference into a static table. May be null.	The matchihng AccessingUser ID. Foreign key to the `AccessingUser` table.
`DomainName`	Alpha-numeric text (maximum 100 characters). May be null.	Domain name of the accessing user.

Property	Attributes	Notes
ExternalAccessing UserID	Unsigned integer (bigint). Mandatory. Database key.	The accessing user id. This is part of the key.
SAMAccountName	Alpha-numeric text (maximum 64 characters). May be null.	SAM account name of the accessing user.
UserName	Alpha-numeric text (maximum 256 characters).	User name of the accessing user.

Inventory Object: ActiveDirectoryComputer

ActiveDirectoryComputer objects are uploaded to the ImportedActiveDirectoryComputer table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedActiveDirectoryComputer table stores the incoming active directory data for computers.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ComputerName	Alpha-numeric text (maximum 64 characters).	The name of the computer. In Windows, this is the NetBIOS name of the local computer, as returned by GetComputerName(). For UNIX, it is the host name of the machine, as returned by gethostname(2).
DomainName	Alpha-numeric text (maximum 100 characters).	The domain name for the computer.
GUID	A universally unique identifier. Mandatory. Database key.	The GUID of the computer.
SID	Alpha-numeric text (maximum 256 characters). May be null.	The SID of the computer.

Inventory Object: ActiveDirectoryDomain

`ActiveDirectoryDomain` objects are uploaded to the `ImportedActiveDirectoryDomain` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedActiveDirectoryDomain` table stores the incoming active directory domains for a connection source.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`DomainFQDN`	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	The fully qualified name domain name of the AD domain
`FlatName`	Alpha-numeric text (maximum 32 characters).	The AD domain flat name
`LastADImportTime`	Date/time field.	The last time the AD data was imported

Inventory Object: ActiveDirectoryExternalMember

`ActiveDirectoryExternalMember` objects are uploaded to the `ImportedActiveDirectoryExternalMember` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedActiveDirectoryExternalMember` table stores the incoming active directory data for external AD member objects.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ParentGroupGUID`	A universally unique identifier. Mandatory. Database key.	The parent AD group GUID.
`SID`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The SID of the member object.

Inventory Object: ActiveDirectoryGroup

`ActiveDirectoryGroup` objects are uploaded to the `ImportedActiveDirectoryGroup` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedActiveDirectoryGroup` table stores the incoming active directory data for a connection source.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`DomainName`	Alpha-numeric text (maximum 100 characters).	The domain name for the user.
`GUID`	A universally unique identifier. Mandatory. Database key.	The GUID of the AD group.
`Name`	Alpha-numeric text (maximum 128 characters). May be null.	The AD group name
`SID`	Alpha-numeric text (maximum 256 characters). May be null.	The SID of the AD group.

Inventory Object: ActiveDirectoryMember

`ActiveDirectoryMember` objects are uploaded to the `ImportedActiveDirectoryMember` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedActiveDirectoryMember` table stores the incoming active directory data for AD member objects.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`GUID`	A universally unique identifier. Mandatory. Database key.	The GUID of the member object.
`ParentGroupGUID`	A universally unique identifier. Mandatory. Database key.	The parent AD group GUID.

Inventory Object: ActiveDirectoryUser

`ActiveDirectoryUser` objects are uploaded to the `ImportedActiveDirectoryUser` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedActiveDirectoryUser` table stores the incoming active directory data for users.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`DomainName`	Alpha-numeric text (maximum 100 characters).	The domain name for the user.
`GUID`	A universally unique identifier. Mandatory. Database key.	The GUID of the user.
`SAMAccountName`	Alpha-numeric text (maximum 20 characters).	The user name.
`Sid`	Alpha-numeric text (maximum 256 characters). May be null.	The Sid for the user.

Inventory Object: ActiveSyncDevice

`ActiveSyncDevice` objects are uploaded to the `ImportedActiveSyncDevice` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedActiveSyncDevice` table stores details of ActiveSync partnerships. A partnership is a user/device pair, so there may be multiple rows for one device.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ActiveSyncID	Alpha-numeric text (maximum 512 characters). Mandatory. Database key.	The EASIdentity presented by the source, a combination of the AD user and the unique device ID. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
DeviceID	Alpha-numeric text (maximum 100 characters). May be null.	The unique device identifier.
DeviceModel	Alpha-numeric text (maximum 100 characters). May be null.	The device model.
DeviceOS	Alpha-numeric text (maximum 100 characters). May be null.	The device operating system.
DeviceType	Alpha-numeric text (maximum 50 characters). May be null.	The device type.
DeviceUserAgent	Alpha-numeric text (maximum 100 characters). May be null.	The device user agent; an ActiveSync client-specific value that may identify the device type.
Domain	Alpha-numeric text (maximum 100 characters). May be null.	The domain of the device. This may be a flat name or FQDN.
EmailAddress	Alpha-numeric text (maximum 256 characters). May be null.	The user's primary email address.
ExchangeServer	Alpha-numeric text (maximum 256 characters). May be null.	The source exchange server for this information.
IMEI	Alpha-numeric text (maximum 256 characters). May be null.	IMEI (International Mobile Equipment Identity) is a 15- or 17-digit code that uniquely identifies mobile phone sets. Leave blank (null) for other device types.

Property	Attributes	Notes
LastSuccessSync	Date/time field. May be null.	The last successful sync time for this partnership, in UTC.
LastSyncAttemptTime	Date/time field. May be null.	The last attempted sync time for this partnership, in UTC.
PhoneNumber	Alpha-numeric text (maximum 128 characters). May be null.	The phone number of the device. Used for mobile devices.
UserDisplayName	Alpha-numeric text (maximum 256 characters). May be null.	The AD user display name.
WhenCreatedUTC	Date/time field. May be null.	The date/time this partnership was created, in UTC.

Inventory Object: ClientAccessEvidence

ClientAccessEvidence objects are uploaded to the ImportedClientAccessEvidence table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedClientAccessEvidence table holds all of the client access evidence which has been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
Edition	Alpha-numeric text (maximum 50 characters). May be null.	The edition of the installed product.
ExternalAccessEvidenceID	Unsigned integer (bigint). Mandatory. Database key.	The identifier of the client access evidence.
ProductName	Alpha-numeric text (maximum 256 characters). May be null.	The name of the product being accessed by user or computer. This may include version and edition too.
UALRoleGUID	A universally unique identifier. May be null.	The UAL role GUID of the product being accessed by user or computer. This is used when retrive data using UAL
UALRoleName	Alpha-numeric text (maximum 256 characters). May be null.	The UAL role name of the product being accessed by user or computer. This is used when retrive data using UAL.

Property	Attributes	Notes
Version	Alpha-numeric text (maximum 72 characters). May be null.	The version of the installed product.

Inventory Object: ClientAccessEvidenceMapping

ClientAccessEvidenceMapping objects are uploaded to the ImportedClientAccessEvidenceMapping table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedClientAccessEvidenceMapping is the mapping table for imported access evidence and access evidence

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
AccessEvidenceID	A numeric reference into a static table. Mandatory. Database key.	Access evidend id. Foreign key to AccessEvidence table.
ExternalAccess EvidenceID	Unsigned integer (bigint). Mandatory. Database key.	External Access evidend id. Foreign key to ImportedClientAccessedAccessEvidence table.

Inventory Object: ClientAccessedAccessEvidence

ClientAccessedAccessEvidence objects are uploaded to the ImportedClientAccessedAccessEvidence table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedClientAccessedAccessEvidence table holds a record of the installer evidence that has been installed on a computer from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ClientAccessSource	Alpha-numeric text (maximum 100 characters). Default: 'Unknown'. Mandatory. Database key.	Referencing to the client access source type.
ExternalAccess EvidenceID	Unsigned integer (bigint). Mandatory. Database key.	Access evidence id .Foreign key to the ImportedClientAccessEvidence table.

Property	Attributes	Notes
ExternalAccessing DeviceID	Unsigned integer (bigint). Mandatory. Database key.	Accessing computer id .Foreign key to the ImportedAccessingDevice table ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
ExternalAccessing UserID	Unsigned integer (bigint). Mandatory. Database key.	Accessing userid. Foreign key to the ImportedAccessingUser table ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
ExternalServer ComputerID	Unsigned integer (bigint). Mandatory. Database key.	Server computer id .Foreign key to the ImportedComputer table. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
ImportedClient AccessedAccess EvidenceID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the installer evidence.

Inventory Object: ClientAccessedAccessOccurrence

`ClientAccessedAccessOccurrence` objects are uploaded to the `ImportedClientAccessedAccessOccurrence` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedClientAccessedAccessOccurrence` table holds the access information of device or user

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccessCount`	Unsigned integer (`int`). Default: 1.	Number of access frequency for given date
`AccessDate`	Date/time field. May be null.	The access date.
`ImportedClient AccessedAccess EvidenceID`	Unsigned integer (`bigint`). Mandatory. Database key.	Access evidence id. Foreign key to the `ImportedClientAccessedAccessEvidence` table.
`InventoryDate`	Date/time field.	Date on which inventory occurance was recorded.
`LicenseDate`	Date/time field. Mandatory. Database key.	Date which will be used for licensing purpose.

Inventory Object: Cluster

`Cluster` objects are uploaded to the `ImportedCluster` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedCluster` table holds all of the clusters which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ClusterTypeID`	A numeric reference into a static table. Default: 1.	The type of cluster.
`DPM`	Boolean (0 or 1). May be null.	Whether Distributed Power Management (DPM) is enabled
`DRS`	Boolean (0 or 1). May be null.	Whether Distributed Resource Scheduler (DRS) is enabled

Property	Attributes	Notes
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.
InventoryDate	Date/time field. May be null.	The date the cluster last had inventory reported.
Name	Alpha-numeric text (maximum 256 characters).	The user-visible name of the cluster.
Namespace	Alpha-numeric text (maximum 256 characters). May be null.	The name of the domain/datacenter containing the cluster.

Inventory Object: ClusterGroup

ClusterGroup objects are uploaded to the ImportedClusterGroup table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedClusterGroup table holds all of the group objects defined on clusters which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ClusterExternalID	Unsigned integer (bigint). Mandatory. Database key.	The unique identifier for the imported cluster.
ClusterID	A numeric reference into a static table. May be null.	The assigned identifier for this cluster group.

Property	Attributes	Notes
ClusterTypeID	A numeric reference into a static table. Default: 3.	Foreign key to the ClusterType table.
ExternalID	Unsigned integer (bigint). Mandatory. Database key.	The unique identifier for this imported cluster group. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
Name	Alpha-numeric text (maximum 256 characters).	The name of the cluster group.

Inventory Object: ClusterGroupMember

ClusterGroupMember objects are uploaded to the ImportedClusterGroupMember table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedClusterGroupMember table holds all of the group memberships defined on clusters which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ClusterGroupExternalID	Unsigned integer (bigint). Mandatory. Database key.	The unique identifier for the imported cluster group.

Property	Attributes	Notes
ComputerExternalID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the external computer which is a member of the group. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: ClusterHostAffinityRule

ClusterHostAffinityRule objects are uploaded to the ImportedClusterHostAffinityRule table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedClusterHostAffinityRule table holds all of the host affinity rules for a cluster which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ClusterExternalID	Unsigned integer (bigint). Mandatory. Database key.	The unique identifier for the imported cluster.
ClusterHostAffinity RuleTypeID	A numeric reference into a static table. Default: 1.	A unique identifier indicating a type of Cluster Host Affinity Rule.
ClusterHostGroup ExternalID	Unsigned integer (bigint). Mandatory. Database key.	The unique identifier for the imported cluster host group.
ClusterVMGroup ExternalID	Unsigned integer (bigint). Mandatory. Database key.	The unique identifier for the imported cluster VM group.
Name	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The name of the cluster group.

Inventory Object: ClusterNode

`ClusterNode` objects are uploaded to the `ImportedClusterNode` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedClusterNode` table holds all of the cluster nodes which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ClusterExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The unique identifier for the imported cluster.
`ClusterNodeTypeID`	A numeric reference into a static table. Default: 1.	Foreign key to the `ClusterNodeType` table.
`ComputerExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the external computer which is a member of the cluster. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: Computer

`Computer` objects are uploaded to the `ImportedComputer` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedComputer` table holds all of the computers which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`CalculatedUser`	Alpha-numeric text (maximum 128 characters). May be null.	The domain/SAMAccountName of the calculated user. Some inventory systems calculate the user who owns a computer. For example, it might be the user who, over the last ten logins, logged in most often.

Property	Attributes	Notes
ChassisType	Alpha-numeric text (maximum 128 characters). May be null.	The type of case of the computer. The value must be a (case insensitive) exact match for one of the values shown. Note that some license types use this information to optimize the licensing position, particularly with desktop and laptop computers.
ComplianceComputerTypeID	Unsigned integer (int). May be null.	If you know that the computer is a virtual machine or VM host, record that data here. If you are unsure, leave this cell empty (NULL): this allows the system to infer the computer type (for example, a computer with VMs linked to it is inferred to be a VM host). If data comes from multiple inventory sources, leaving this value as null also allows the value to be inserted from another source. So, unless there is a very good reason, do not just specify 'Computer', but allow the inference rules to help.
ComputerName	Alpha-numeric text (maximum 256 characters). May be null.	The name of the computer. In Windows, this is the NetBIOS name of the local computer, as returned by GetComputerName(). For UNIX, it is the host name of the machine, as returned by gethostname(2).
Domain	Alpha-numeric text (maximum 100 characters). May be null.	The domain of the computer.
EmailAddress	Alpha-numeric text (maximum 256 characters). May be null.	The email address associated with the device. Typically used for mobile devices.
ExternalID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the computer. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
FirmwareSerialNumber	Alpha-numeric text (maximum 100 characters). May be null.	Serial number in the system firmware such as BIOS, EEPROM etc.
HardwareInventoryDate	Date/time field. May be null.	The date (and optionally time) when the hardware was last inventoried. For automated/scheduled data uploads through an inventory beacon, make sure that inventory dates are kept current, as they are used to report out-of-date inventory sources. For a one-time upload to the central application server, leave inventory dates empty (null). At each import from the saved file, the import date is used as the inventory date, which prevents the inventory becoming stale. Notice that this value is not available in the web interface.
HostID	Alpha-numeric text (maximum 100 characters). May be null.	The HostID hardware property for the server hosting this machine partition (when inventorying a machine partition such as Solaris Zone, AIX lPar, HP-UX nPar/vPar).
HostIdentifyingNumber	Alpha-numeric text (maximum 128 characters). May be null.	Virtual hosts may have an identifier that is unique only across that hardware model. It is less unique than the true hardware serial number, for example.
HostType	Alpha-numeric text (maximum 128 characters). May be null.	The type of the physical host computer.
ILMTAgentID	Unsigned integer (bigint). May be null.	The unique ID used by the IBM License Metric Tool (ILMT) inventory agent on this device, if the inventory source is aware of this value. This can be used to track a computer over time and can be used to socialize different inventory sources. Currently the ILMT and ManageSoft inventory adapters report this value.
IMEI	Alpha-numeric text (maximum 256 characters). May be null.	IMEI (International Mobile Equipment Identity) is a 15- or 17-digit code that uniquely identifies mobile phone sets. Leave blank (null) for other device types.
IPAddress	Alpha-numeric text (maximum 256 characters). May be null.	The IP address of the computer.

Property	Attributes	Notes
IgnoredDueToLicense	Boolean (0 or 1). Default: 0.	True if this machine is not imported into compliance computer table due to license limitation
IncompleteRecord	Boolean (0 or 1). May be null.	Used to identify records which do not have all information specified. Primarily used for ManageSoft source connections where the domain name was not reliably reported.
InventoryAgent	Alpha-numeric text (maximum 128 characters).	The name of the person or tool that performed the last inventory. For imported spreadsheets, you may wish to include the name of the person preparing the data, in case there is subsequent follow-up required.
InventoryDate	Date/time field. May be null.	The date the computer last had inventory reported.
IsDuplicate	Boolean (0 or 1). Default: 0.	Used to identify that imported computer is a duplicate of another, whereby a new computer will not created.
IsRemoteACLDevice	Boolean (0 or 1). Default: 0.	Used to determine if the current record is a remote ACL based device.
LastLoggedOnUser	Alpha-numeric text (maximum 128 characters). May be null.	The DOMAIN/SAMAccountName of the user last logged onto the computer.
LastSuccessful InventoryDate	Date/time field. May be null.	For incremental imports, this represents the inventory date of the computer in the source at the time this record was last successfully imported. If the import procedure has failed, this may be different to the inventory date. At the end of a successful incremental import, this value is updated to match the inventory date. If no value is present in this field, either there has not been a successful import of this computer or the reader for this record is not using an incremental update model.
LegacySerialNo	Alpha-numeric text (maximum 100 characters). May be null.	A previous serial number of this computer that can also be used for matching.
MACAddress	Alpha-numeric text (maximum 256 characters). May be null.	The MAC address of the computer.

Property	Attributes	Notes
MDScheduleContainsPVUScan	Boolean (0 or 1). Default: 0. May be null.	Does this managed device include an event in its current schedule for running extra IBM PVU hardware scans.
MDScheduleGenerated Date	Date/time field. May be null.	The last time the managed device schedule was regenerated.
MachineID	Alpha-numeric text (maximum 100 characters). May be null.	For AIX, it is the System ID. For HP-UX, it is the Machine/Software ID. It is unset for other platforms.
Manufacturer	Alpha-numeric text (maximum 128 characters). May be null.	The manufacturer of the computer hardware.
MaxClockSpeed	Unsigned integer (int). May be null.	The maximum clock speed of the fastest processor in the computer.
ModelNo	Alpha-numeric text (maximum 128 characters). May be null.	The model number of the computer.
NumberOfCores	Unsigned integer (int). May be null.	The number of cores in the computer.
NumberOfDisplay Adapters	Unsigned integer (int). May be null.	The number of graphics cards in the computer.
NumberOfHardDrives	Unsigned integer (int). May be null.	The number of hard drives in the computer.
NumberOfLogical Processors	Unsigned integer (int). May be null.	The number of logical processors in the computer.
NumberOfNetworkCards	Unsigned integer (int). May be null.	The number of network cards in the computer.
NumberOfProcessors	Unsigned integer (int). May be null.	The number of processors in the computer.
NumberOfSockets	Unsigned integer (int). May be null.	The number of sockets in the computer.
OperatingSystem	Alpha-numeric text (maximum 128 characters). May be null.	The operating system of the computer.
PartialNumberOf Processors	Fractional number (float). May be null.	The fractional processor count available to this computer.

Property	Attributes	Notes
PhoneNumber	Alpha-numeric text (maximum 128 characters). May be null.	The phone number of the device. Used for mobile devices.
ProcessorType	Alpha-numeric text (maximum 256 characters). May be null.	The type of processor in the computer.
SerialNo	Alpha-numeric text (maximum 100 characters). May be null.	The serial number of the computer.
ServicePack	Alpha-numeric text (maximum 128 characters). May be null.	The service pack installed for the operating system.
ServicesInventoryDate	Date/time field. May be null.	The date when services (for example, Oracle) were last scanned on this computer. For automated/scheduled data uploads through an inventory beacon, make sure that inventory dates are kept current, as they are used to report out-of-date inventory sources. For a one-time upload to the central application server, leave inventory dates empty (null). At each import from the saved file, the import date is used as the inventory date, which prevents the inventory becoming stale.
TotalDiskSpace	Unsigned integer (bigint). May be null.	The total size of all hard drives in the computer.
TotalMemory	Unsigned integer (bigint). May be null.	The total RAM in the computer, in bytes.
UUID	A universally unique identifier. May be null.	The BIOS UUID of the computer.
UntrustedSerialNo	Boolean (0 or 1). Default: 0.	Is this computer known to have a serial number from a data source that should not be trusted.

Inventory Object: ComputerCustomProperty

`ComputerCustomProperty` objects are uploaded to the `ImportedComputerCustomProperty` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedComputerCustomProperty` table is used by the importer to import custom properties for computers.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier, in the source connection, of the computer that this property belongs to.
`PropertyNameID`	Unsigned integer (`int`). Mandatory. Database key.	The identifier for custom property in the `ImportedCustomPropertyName` table.
`PropertyValue`	Alpha-numeric text (maximum 256 characters).	The value of the custom property.

Inventory Object: ConsolidatedAccessEvidence

`ConsolidatedAccessEvidence` objects are uploaded to the `ConsolidatedAccessEvidence` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

ConsolidatedAccessEvidence provides a simpler interface to specify client access happening on application installed on server computers. It combines the server computer, and its access evidence details into a single row.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccessCount`	Unsigned integer (`int`). Default: 1. May be null.	Number of times the product was accessed on the given access date.
`AccessDate`	Date/time field. Mandatory. Database key.	The access date of the access evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
AccessingDevice ComputerName	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	IP Address of the device accessing the product. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
AccessingDeviceDomain	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	Domain name of the device accessing the product. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
AccessingDeviceIP Address	An ASCII string of alphanumeric characters and punctuation (length 256 characters). Mandatory. Database key.	IP Address of the accessing device. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
AccessingDevice SerialNo	Alpha-numeric text (maximum 100 characters). May be null.	Serial number of the device accessing the product.

Property	Attributes	Notes
AccessingUser	Alpha-numeric text (maximum 128 characters). Mandatory. Database key.	The DOMAIN/SAMAccountName of the user accessing the product. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
ClientAccessSource	Alpha-numeric text (maximum 100 characters). Default: Manual. May be null.	The source type of the access evidence.
ComputerID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the computer. It must match the ComputerID from the Computer spreadsheet or the row will be ignored.
Edition	Alpha-numeric text (maximum 50 characters). Default: . Mandatory. Database key.	The edition of the software as reported by the access evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
InventoryDate	Date/time field. Default: getdate(). May be null.	The date (and optionally time) the access evidence record was inventoried.
ProductName	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The product name of the software as reported by the access evidence.

Property	Attributes	Notes
Version	Alpha-numeric text (maximum 72 characters). Default: . Mandatory. Database key.	The version of the software as reported by the access evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: ConsolidatedCluster

ConsolidatedCluster objects are uploaded to the ConsolidatedCluster table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The Cluster spreadsheet provides a simple interface for defining server clustering. It is useful when combined with the ClusterGroup and ClusterHostAffinityRule spreadsheets.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ClusterID	Unsigned integer (bigint). Mandatory. Database key.	The unique identifier for this imported cluster. This may be a string or an integer.
ClusterName	Alpha-numeric text (maximum 128 characters).	The name of the cluster in the external cluster management system.
ClusterType	Alpha-numeric text (maximum 128 characters).	The kind of cluster. The value must be an exact case-insensitive match to one of the permitted values.
DPM	Boolean (0 or 1). May be null.	Whether Distributed Power Management (DPM) is enabled on the cluster.
DRS	Boolean (0 or 1). May be null.	Whether Distributed Resource Scheduler (DRS) is enabled on the cluster.
InventoryAgent	Alpha-numeric text (maximum 64 characters). May be null.	The name of the person or tool that performed the last inventory. For imported spreadsheets, you may wish to include the name of the person preparing the data, in case there is subsequent follow-up required.

Property	Attributes	Notes
InventoryDate	Date/time field. May be null.	The date (with optional time) that the cluster last had inventory reported.
Namespace	Alpha-numeric text (maximum 256 characters). May be null.	Where the cluster is contained.

Inventory Object: ConsolidatedClusterGroup

ConsolidatedClusterGroup objects are uploaded to the ConsolidatedClusterGroup table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ClusterGroup spreadsheet uses data from the Cluster spreadsheet and defines groups of servers as well as computers that are members of these groups.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ClusterGroupID	Unsigned integer (bigint). Mandatory. Database key.	The unique identifier for this cluster group. This may be a string or an integer.
ClusterGroupName	Alpha-numeric text (maximum 128 characters). May be null.	The name of the cluster group. Depending on the value of the ClusterGroupType this will be a group of hosts or virtual machines.
ClusterGroupType	Alpha-numeric text (maximum 128 characters).	The kind of cluster included in the group. The value must be an exact case-insensitive match to one of the permitted values.
ClusterID	Unsigned integer (bigint). Mandatory. Database key.	The unique identifier for the imported cluster. This may be a string or an integer and must match a value for the ClusterID in the cluster spreadsheet.
ComputerID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the 'Computer' spreadsheet for a computer which is a member of the group. To identify all the members of the group, repeat as many lines as required in your spreadsheet where the other values in the row are identical, and only the 'ComputerID' value changes. Values in this column must match a ComputerID in the computer spreadsheet or the row will be skipped.

Inventory Object: ConsolidatedClusterHostAffinityRule

`ConsolidatedClusterHostAffinityRule` objects are uploaded to the `ConsolidatedClusterHostAffinityRule` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ClusterHostAffinity spreadsheet defines the groups of virtual machines which may run on groups of host servers.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ClusterHostAffinity RuleType`	Alpha-numeric text (maximum 128 characters).	The type of affinity rule. The value must be an exact case-insensitive match to one of the permitted values.
`ClusterHostGroupName`	Unsigned integer (`bigint`). Mandatory. Database key.	The name of the group of hosts that the ClusterVMGroupName virtual machines may run on.
`ClusterID`	Unsigned integer (`bigint`). Mandatory. Database key.	The unique identifier for the imported cluster, to which this affinity rule applies. This may be a string or an integer and must match a ClusterID from the cluster spreadsheet.
`ClusterVMGroupName`	Unsigned integer (`bigint`). Mandatory. Database key.	The name of the virtual machine group that may run on the ClusterHostGroupName hosts.
`Name`	Alpha-numeric text (maximum 128 characters). May be null.	The name of the cluster host affinity rule.

Inventory Object: ConsolidatedComputer

`ConsolidatedComputer` objects are uploaded to the `ConsolidatedComputer` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

'ConsolidatedComputer' consolidates data for the Computer, VirtualMachine, Domain, User and Cluster objects, providing a simpler way to populate this information. Any spreadsheet row that includes a 'HostComputerID' is making that row a virtual machine, and the import process expects that virtualization data will be provided.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AffinityEnabled`	Boolean (0 or 1). Default: 0.	Set this to `true` (or `1`) if this VM has affinity for its current host (so that it is unable to move to different host computers).
`BIOSUUID`	A universally unique identifier. May be null.	The BIOS UUID of the computer (physical or virtual), as provided by the operating system.
`CPUAffinity`	Alpha-numeric text (maximum 256 characters). May be null.	Contains a comma-separated list of processor numbers (Host Logical Processors) or ranges for which this virtual machine has affinity. Example: `1,3-5,8`
`CPUUsage`	Unsigned integer (`int`). May be null.	The maximum CPU usage of the virtual machine (MHz).
`CalculatedUser`	Alpha-numeric text (maximum 128 characters). May be null.	The domain/SAMAccountName of the calculated user. Some inventory systems calculate the user who owns a computer. For example, it might be the user who, over the last ten logins, logged in most often.
`ChassisType`	Alpha-numeric text (maximum 128 characters). May be null.	The chassis type of the device.
`ClusterID`	Unsigned integer (`bigint`). Mandatory. Database key.	The unique identifier for the cluster containing this computer. This must match the ClusterID used in the Cluster spreadsheet. If both the ClusterID and the ClusterNodeType do not match the data provided in the Cluster spreadsheet then the computer will not be associated with a cluster. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`ClusterNodeType`	Alpha-numeric text (maximum 128 characters). Default: 1. May be null.	The Cluster node type of the computer. Must be a (case insentitive) exact match for one of the values shown. If both the ClusterID and the ClusterNodeType do not match the data provided in the Cluster spreadsheet then the computer will not be associated with a cluster.
`ComplianceComputerType`	Alpha-numeric text (maximum 128 characters). May be null.	If you know that the computer is a virtual machine or VM host, record that data here. If you are unsure, leave this cell empty (NULL): this allows the system to infer the computer type (for example, a computer with VMs linked to it is inferred to be a VM host). If data comes from multiple inventory sources, leaving this value as null also allows the value to be inserted from another source. So, unless there is a very good reason, do not just specify 'Computer', but allow the inference rules to help.
`ComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The unique identifier for a computer (either physical or virtual). This identifier can either be an integer or a string. Keep this consistent across multiple imports: it is used to track the computer over time.
`ComputerName`	Alpha-numeric text (maximum 256 characters).	The name of the computer. In Windows, this is the NetBIOS name of the local computer, as returned by `GetComputerName()`. For UNIX, it is the host name of the machine, as returned by `gethostname(2)`.
`CoreAffinity`	Alpha-numeric text (maximum 256 characters). May be null.	Contains a comma-separated list of core numbers (or ranges) for which this virtual machine has affinity. Cores are numbered sequentially up the sequence of processors. Example: `1,5-8,10`

Property	Attributes	Notes
DomainFlatName	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	The flatname of the domain of the computer. Example: 'mycompany'. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
DomainQualifiedName	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	The fully qualified domain name for the computer. Example: 'prod.mycompany.eu'. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
EmailAddress	Alpha-numeric text (maximum 256 characters). May be null.	The email address associated with the device. Typically used for mobile devices.
FirmwareSerialNumber	Alpha-numeric text (maximum 100 characters). May be null.	The Serial number in the system firmware such as BIOS, EEPROM etc.

Property	Attributes	Notes
HostComputerID	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The ComputerID of the server this virtual machine is hosted on. This may be a string or an integer and must match the ComputerID for another computer in this spreadsheet. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
HostID	Alpha-numeric text (maximum 100 characters). May be null.	The HostID hardware property for the server hosting this machine partition (when inventorying a machine partition such as Solaris Zone, AIX lPar, HP-UX nPar/vPar).
HostIdentifyingNumber	Alpha-numeric text (maximum 128 characters). May be null.	Virtual hosts may have an identifier that is unique only across that hardware model. It is less unique than the true hardware serial number, for example.
HostType	Alpha-numeric text (maximum 128 characters). May be null.	The type (similar to model number) of the host, used for matching.
IMEI	Alpha-numeric text (maximum 256 characters). May be null.	IMEI (International Mobile Equipment Identity) is a 15- or 17-digit code that uniquely identifies mobile phone sets. Leave blank (null) for other device types.
IPAddress	Alpha-numeric text (maximum 256 characters). May be null.	The IP address of the computer in IPv4 or IPv6 format.
InventoryDate	Date/time field. Default: getdate(). May be null.	The date (and optionally time) the computer last had inventory reported. This field is generally used for differential updates (that is, if the date/time has not changed since the previous import, the data record is not imported/updated).

Property	Attributes	Notes
`LastLoggedOnUser`	Alpha-numeric text (maximum 128 characters). Mandatory. Database key.	The DOMAIN/SAMAccountName of the user last logged onto the computer. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`LastLogonDate`	Date/time field. May be null.	The date and time when the user last logged on to the computer.
`MACAddress`	Alpha-numeric text (maximum 256 characters). May be null.	The MAC address of the computer. This may be a comma-separated list if there is more than one active network adapter in the system. Do not include inactive network adapters and network adapters with invalid MAC addresses.
`MachineID`	Alpha-numeric text (maximum 100 characters). May be null.	For AIX, it is the System ID. For HP-UX, it is the Machine/Software ID. It is unset for other platforms.
`Manufacturer`	Alpha-numeric text (maximum 128 characters). May be null.	The manufacturer of the computer.
`MaxClockSpeed`	Unsigned integer (`int`). May be null.	The maximum clock speed of the fastest processor in the computer in kHz. Note that a number of server-based licenses depend on complete details of the processor types, counts and speeds to calculate a correct license position.
`MemoryUsage`	Unsigned integer (`bigint`). May be null.	The maximum memory usage of the virtual machine (bytes).
`ModelNo`	Alpha-numeric text (maximum 128 characters). May be null.	The model number of the computer.

Property	Attributes	Notes
`NumberOfCores`	Unsigned integer (`int`). May be null.	The total number of cores in the computer. If there is more than one physical processor in the computer, then this would be the sum of the core counts for all the processors. For example, in a computer with two quad-core processors, this value would be 8. Note that a number of server-based licenses depend on complete details of the processor types, counts and speeds to calculate a correct license position.
`NumberOfDisplay Adapters`	Unsigned integer (`int`). May be null.	The number of graphics cards in the computer.
`NumberOfHardDrives`	Unsigned integer (`int`). May be null.	The number of physical hard drives in the computer. While the intent is physical drives, often this can end up being the number of disk partitions.
`NumberOfLogical Processors`	Unsigned integer (`int`). May be null.	The number of logical processors in the computer. This is the number of 'execution contexts' the operating system has access to. It will commonly be equivalent to the number processors in a single core, non-multi-threaded processor architecture, to the number of cores in a multi-core single threaded processor architecture, and to the number of threads in a multi-threaded processor architecture. For example, in a two processor, quad-core and hyper-threaded computer, this value would be 16. Note that a number of server-based licenses depend on complete details of the processor types, counts and speeds to calculate a correct license position.
`NumberOfNetworkCards`	Unsigned integer (`int`). May be null.	The number of network cards in the computer.
`NumberOfProcessors`	Unsigned integer (`int`). May be null.	The total number of physical processors (CPU) in the computer. Note that a number of server-based licenses depend on complete details of the processor types, counts and speeds to calculate a correct license position.

Property	Attributes	Notes
NumberOfSockets	Unsigned integer (int). May be null.	The number of physical sockets into which a processor may be placed in the computer. It is rare that an inventory source can know this value. If unset, it is typically approximated by the number of processors.
OperatingSystem	Alpha-numeric text (maximum 128 characters). May be null.	The operating system of the computer. For virtual machines, it is the configured operating system of the guest. Note that this operating system identification is not used for licensing.
PartialNumberOf Processors	Fractional number (float). May be null.	Used in processor-based licensing, this is the non-integer number of cores allocated to this partition or virtual machine. When this property is null, the 'NumberOfCores' is used. Note that a number of server-based licenses depend on complete details of the processor types, counts and speeds to calculate a correct license position.
PhoneNumber	Alpha-numeric text (maximum 128 characters). May be null.	The phone number of the device. Used for mobile devices.
PoolName	Alpha-numeric text (maximum 100 characters). May be null.	The name of the pool that the virtual machine belongs to.
PoolType	Alpha-numeric text (maximum 100 characters). May be null.	The type of the pool that the virtual machine belongs to.
ProcessorType	Alpha-numeric text (maximum 256 characters). May be null.	The descriptive string of the processor(s) in the computer. This may be a comma-separated list in the case where there is more than one physical processor in the system. Note that a number of server-based licenses depend on complete details of the processor types, counts and speeds to calculate a correct license position.
SerialNo	Alpha-numeric text (maximum 100 characters). May be null.	The serial number of the computer.
ServicePack	Alpha-numeric text (maximum 128 characters). May be null.	The service pack installed for the operating system.

Property	Attributes	Notes
`TotalDiskSpace`	Unsigned integer (`bigint`). May be null.	The total size of all hard drives in the computer in bytes. Note that this can be a very large number on modern systems. The maximum value for a bigint is 9,223,372,036,854,775,807, which can represent about 9.2 exabyte. While in practice it is unlikely that this size of storage capacity is reached for a single system, some systems can end up with large values through virtualized drives. Therefore, it is worth considering capping values when calculating total disk space, particularly when converting values from kilobytes or megabytes to bytes.
`TotalMemory`	Unsigned integer (`bigint`). May be null.	The total RAM in the computer, in bytes.
`VMEnabledState`	Alpha-numeric text (maximum 128 characters). Default: 4. May be null.	The operational state of the virtual machine. If present, the value must be a (case insensitive) exact match to one of the values shown.
`VMLocation`	Alpha-numeric text (maximum 256 characters). May be null.	Location of the virtual machine on the file system.
`VirtualMachineType`	Alpha-numeric text (maximum 100 characters). May be null.	The type of the virtual machine. If present, the value must be a (case insensitive) exact match to one of the values shown.
`VirtualMachineUUID`	Alpha-numeric text (maximum 256 characters). May be null.	The unique identifier of the virtual machine provided by the virtualization infrastructure. (This may have the same value as the 'BIOSUUID', or have byte order reversed, or be altogether different.)

Inventory Object: ConsolidatedFileEvidence

`ConsolidatedFileEvidence` objects are uploaded to the `ConsolidatedFileEvidence` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

ConsolidatedFileEvidence provides a simpler interface to specify files and their usage on computers. It combines the computer, file evidence and usage details into a single row.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccessMode`	Alpha-numeric text (maximum 128 characters). Default: 1. Mandatory. Database key.	The access mode of the file evidence. Leave this blank unless this row is a virtualized application. In that case choose one of the values below that matches your application or desktop virtualization infrastructure. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`Company`	Alpha-numeric text (maximum 100 characters). Default: . Mandatory. Database key.	The company in the file header. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer. It must match the ComputerID from the Computer spreadsheet or the row will be ignored.

Property	Attributes	Notes
`Description`	Alpha-numeric text (maximum 200 characters). Default: . Mandatory. Database key.	The description in the file header. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`FileName`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The name of the file used as evidence of software installation. For unix operating systems include the full path in the file name, including the opening '/'. For Windows operating systems the file path is specified in the FilePath column and this column must only contain the file name.
`FilePath`	Alpha-numeric text (maximum 400 characters). May be null.	The path of the file used as evidence of software installation.
`FileSize`	Unsigned integer (`int`). Default: 0. Mandatory. Database key.	The size of the file in bytes. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
FileVersion	Alpha-numeric text (maximum 100 characters). Default: . Mandatory. Database key.	The version number of the file used as evidence of software installation. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
Language	Alpha-numeric text (maximum 200 characters). May be null.	The language in the file header.
LastUsedDate	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The last used date of the usage.
NumberOfSessions	Unsigned integer (bigint). May be null.	The number of sessions that the file evidence was in use by the user specified in the UserID column during the usage tracking period. If multiple users used the same application on the computer, create one row for each user with usage.
ProductName	Alpha-numeric text (maximum 200 characters). May be null.	The product name in the file header.
ProductVersion	Alpha-numeric text (maximum 200 characters). May be null.	The product version number in the file header.
StartDate	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The start date of the usage.

Property	Attributes	Notes
UserID	Unsigned integer (bigint). Mandatory. Database key.	The DOMAIN/SAMAccountName for the user that the file evidence was used by. If this software was used by multiple users, create one row for each user of the software on the computer. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: ConsolidatedInstallerEvidence

ConsolidatedInstallerEvidence objects are uploaded to the ConsolidatedInstallerEvidence table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

ConsolidatedInstallerEvidence provides a simpler interface to specify installed applications and their usage on computers. It combines the computer, installer evidence and usage details into a single row.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
AccessMode	Alpha-numeric text (maximum 128 characters). Default: 1. Mandatory. Database key.	The access mode of the installer evidence. Leave this blank unless this row is a virtualized application. In that case choose one of the values below that matches your application or desktop virtualization infrastructure. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`ComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer. It must match the ComputerID from the Computer spreadsheet or the row will be ignored.
`DatabaseName`	Unsigned integer (`bigint`). Mandatory. Database key.	If this installer evidence is an Oracle Database, then this field specifies the name of the database. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`DiscoveryDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The date that the installer evidence was first seen.
`DisplayName`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The display name of the software as reported by the installer evidence.
`Evidence`	Alpha-numeric text (maximum 32 characters). Default: . Mandatory. Database key.	Identifier for the type of installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`InstallDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The install date of the installer evidence.

Property	Attributes	Notes
`InstanceName`	Unsigned integer (`bigint`). Mandatory. Database key.	If this installer evidence is an Oracle d atabase, then this field specifies the name of the database instance. If there are multiple instances, create a row for each instance in this spreadsheet. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`LastUsedDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The last used date of the usage.
`NumberOfSessions`	Unsigned integer (`bigint`). May be null.	The number of sessions that the installer evidence was in use by the user specified in the UserID column during the usage tracking period. If multiple users used the same application on the computer, create one row for each user with usage.
`ProductCode`	Alpha-numeric text (maximum 55 characters). May be null.	The product code of the evidence. This is usually the MSI product code.
`Publisher`	Alpha-numeric text (maximum 200 characters). Default: . Mandatory. Database key.	The publisher of the software as reported by the installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
StartDate	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The start date of the usage.
UserID	Unsigned integer (bigint). Mandatory. Database key.	The DOMAIN/SAMAccountName for the user that the installer evidence was used by. If this software was used by multiple users, create one row for each user of the software on the computer. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
Version	Alpha-numeric text (maximum 72 characters). Default: . Mandatory. Database key.	The version of the software as reported by the installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: ConsolidatedOracleDatabaseUser

`ConsolidatedOracleDatabaseUser` objects are uploaded to the `ConsolidatedOracleDatabaseUser` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

ConsolidatedOracleDatabaseUser provides a list of the users for each Oracle Database instance.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccessMode`	Alpha-numeric text (maximum 128 characters). Default: 1. Mandatory. Database key.	The access mode of the installer evidence. Leave this blank unless this row is a virtualized application. In that case choose one of the values below that matches your application or desktop virtualization infrastructure. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`AccountStatus`	Alpha-numeric text (maximum 256 characters). May be null.	The current status of the end-user account.
`ComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer. It must match the ComputerID from the Computer spreadsheet or the row will be ignored.
`CreationDate`	Date/time field. May be null.	The date and time when the end-user was created.
`DatabaseName`	Unsigned integer (`bigint`). Mandatory. Database key.	This field specifies the name of the database. It must match a row in the InstallerEvidence spreadsheet for the same ComputerID or this row will be skipped.
`DefaultTablespace`	Alpha-numeric text (maximum 256 characters). May be null.	The default tablespace for an Oracle end-user.

Property	Attributes	Notes
DisplayName	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The display name of the software as reported by the installer evidence. It must match a row in the InstallerEvidence spreadsheet for the same ComputerID, Version, Publisher, DatabaseName and InstanceName or this row will be skipped.
Evidence	Alpha-numeric text (maximum 32 characters). Default: . Mandatory. Database key.	Identifier for the type of installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
InstanceName	Unsigned integer (bigint). Mandatory. Database key.	This field specifies the name of the database instance. If there are multiple instances, create a row for each instance in this spreadsheet. It must match a row in the InstallerEvidence spreadsheet for the same ComputerID and DatabaseName or this row will be skipped.
LastLogonDate	Date/time field. May be null.	The date and time when the end-user last logged on to the computer.
Name	Alpha-numeric text (maximum 256 characters).	The name of the user.
Publisher	Alpha-numeric text (maximum 200 characters). Default: . Mandatory. Database key.	The publisher of the software as reported by the installer evidence. It must match a row in the InstallerEvidence spreadsheet for the same ComputerID, DisplayName, Version, DatabaseName and InstanceName or this row will be skipped.
TempTablespace	Alpha-numeric text (maximum 256 characters). May be null.	The temporary tablespace for an Oracle end-user.
UserID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the instance end-user. This may be an integer or a string.

Property	Attributes	Notes
Version	Alpha-numeric text (maximum 72 characters). Default: . Mandatory. Database key.	The version of the software as reported by the installer evidence. It must match a row in the InstallerEvidence spreadsheet for the same ComputerID, DisplayName, Publisher, DatabaseName and InstanceName or this row will be skipped.

Inventory Object: ConsolidatedRemoteAccessFile

ConsolidatedRemoteAccessFile objects are uploaded to the ConsolidatedRemoteAccessFile table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The spreadsheet consolidates ImportedRemoteUserToApplicationAccess, ImportedFileEvidence and ImportedUser tables.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
AccessMode	Alpha-numeric text (maximum 128 characters). Default: 1. Mandatory. Database key.	The AccessMode states how an application has been accessed. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
Company	Alpha-numeric text (maximum 100 characters). Default: . Mandatory. Database key.	The company in the file header. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`Description`	Alpha-numeric text (maximum 200 characters). Default: . Mandatory. Database key.	The description in the file header. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`FileName`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The name of the file used as evidence of software installation. For unix operating systems include the full path in the file name, including the opening '/'. For Windows operating systems the file path is specified in the FilePath column and this column must only contain the file name.
`FilePath`	Alpha-numeric text (maximum 400 characters). May be null.	The path of the file used as evidence of software installation.
`FileSize`	Unsigned integer (`int`). Default: 0. Mandatory. Database key.	The size of the file in bytes. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
FileVersion	Alpha-numeric text (maximum 100 characters). Default: . Mandatory. Database key.	The version number of the file used as evidence of software installation. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
Language	Alpha-numeric text (maximum 200 characters). May be null.	The language in the file header.
ProductName	Alpha-numeric text (maximum 200 characters). May be null.	The product name in the file header.
ProductVersion	Alpha-numeric text (maximum 200 characters). May be null.	The product version number in the file header.
ServerID	Unsigned integer (bigint). Mandatory. Database key.	This is the ComputerID of the server that publishes this virtual application. The ComputerID must match a computer from the Computer spreadsheet, and that computer must have an installation of the application this file is part of. If the server does not have an installation of an appropriate application then the user will not be shown as having access to that application. This is a mandatory field.
UserID	Unsigned integer (bigint). Mandatory. Database key.	The fully qualified name of the user.

Inventory Object: ConsolidatedRemoteAccessInstaller

`ConsolidatedRemoteAccessInstaller` objects are uploaded to the `ConsolidatedRemoteAccessInstaller` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The spreadsheet consolidates ImportedRemoteUserToApplicationAccess, ImportedInstallerEvidence and ImportedUser tables.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AccessMode`	Alpha-numeric text (maximum 128 characters). Default: 1. Mandatory. Database key.	The AccessMode states how an application has been accessed. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`DisplayName`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The display name of the software as reported by the installer evidence and is part of the unique identifier for installer evidence.
`Evidence`	Alpha-numeric text (maximum 32 characters). Default: . Mandatory. Database key.	The evidence type of the software as reported by the installer evidence and is part of the unique identifier for installer evidence.
`ProductCode`	Alpha-numeric text (maximum 55 characters). May be null.	The product code of the evidence. This is usually the MSI product code and is not part of the unique identifier.
`Publisher`	Alpha-numeric text (maximum 200 characters). Default: . Mandatory. Database key.	Publishers of software applications (for example, "Microsoft").
`UserID`	Unsigned integer (`bigint`). Mandatory. Database key.	The DOMAIN\SAMAccountName of the user.

Property	Attributes	Notes
Version	Alpha-numeric text (maximum 72 characters). Default: . Mandatory. Database key.	The version of the software as reported by the installer evidence and is part of the unique identifier for installer evidence.

Inventory Object: ConsolidatedVMPool

ConsolidatedVMPool objects are uploaded to the ConsolidatedVMPool table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The VMPool spreadsheet provides a simple method to associate virtual machines with groups (pools) on their host.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
HostComputerID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the computer which is hosting the pool. The HostComputerID should match the ComputerID in the Computer spreadsheet. Otherwise the record will be ignored.
NumberOfCores	Fractional number (float). May be null.	The number of cores in this pool.
NumberOfProcessors	Fractional number (float). May be null.	The number of processors in this pool.
ObjectType	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The type of pool. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
ParentName	Alpha-numeric text (maximum 100 characters). May be null.	The name of the parent pool.
ParentObjectType	Alpha-numeric text (maximum 256 characters). May be null.	The type of pool of the parent.

Property	Attributes	Notes
PoolFriendlyName	Alpha-numeric text (maximum 256 characters).	The friendly name of the pool.
PoolName	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	The name of the pool.

Inventory Object: ConsolidatedWMIEvidence

ConsolidatedWMIEvidence objects are uploaded to the ConsolidatedWMIEvidence table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

ConsolidatedWMIEvidence provides a simpler interface to specify Windows Management Instrumentation (WMI) properties on computers. Other Web-Based Enterprise Management (WBEM) properties are supported from Unix computers as well. The most important data to provide in this spreadsheet is operating system installs. The 'Win32_OperatingSystem' class and the 'Name' property contains this data.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ClassName	Alpha-numeric text (maximum 50 characters). Mandatory. Database key.	The WMI class name of the evidence. An example is 'Win32_OperatingSystem'.
ComputerID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the computer. It must match the ComputerID from the Computer spreadsheet or the row will be ignored.

Property	Attributes	Notes
InstanceName	Alpha-numeric text (maximum 256 characters). Default: . Mandatory. Database key.	The name of the WMI class instance. This is important when there a multiple instances of a WMI class on a computer. An example is the 'Win32_VideoController' class that may have many instances with the same properties. In this case you need to specify the name of the instance here, 'Intel(R) HD Graphics Family' or 'NVIDIA Quadro K2100M' for example. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
PropertyName	Alpha-numeric text (maximum 50 characters). Mandatory. Database key.	The WMI property name of the WMI evidence. An example is 'Name'.
PropertyValue	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The value of the property of the WMI evidence. An example is 'Microsoft Windows 7 Enterprise'

Inventory Object: Domain

Domain objects are uploaded to the ImportedDomain table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedDomain table holds all of the domains which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ComplianceDomainID	Unsigned integer (int). May be null.	Identifier of the domain in the ComplianceDomain table that this imported domain links to. This is populated as part of the import process and does not need to be provided by the source connections.

Property	Attributes	Notes
`FlatName`	Alpha-numeric text (maximum 200 characters). Mandatory. Database key.	The flat name of the domain. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`QualifiedName`	Alpha-numeric text (maximum 200 characters). Mandatory. Database key.	The fully qualified name of the domain. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: EvidenceAttribute

`EvidenceAttribute` objects are uploaded to the `ImportedEvidenceAttribute` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedEvidenceAttribute` table holds all of the instance attributes from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
AttributeID	Unsigned integer (int). Mandatory. Database key.	The identifier used in the source connection for the instance attribute. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
AttributeName	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The name of the instance attribute. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: FileEvidence

FileEvidence objects are uploaded to the ImportedFileEvidence table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedFileEvidence table holds all of the file evidence which has been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
AccessModeID	Unsigned integer (int). Default: 1. May be null.	The access mode ID of the file evidence.
Company	Alpha-numeric text (maximum 100 characters). May be null.	The company in the file header.
Description	Alpha-numeric text (maximum 200 characters). Default: ''.	The description in the file header.

Property	Attributes	Notes
ExternalFileID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the file evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
FileName	Alpha-numeric text (maximum 256 characters). May be null.	The name of the file used as evidence of software installation.
FilePath	Alpha-numeric text (maximum 400 characters). May be null.	The path of the file used as evidence of software installation.
FileSize	Unsigned integer (int). May be null.	The size of the file.
FileVersion	Alpha-numeric text (maximum 100 characters). May be null.	The version number of the file used as evidence of software installation.
Language	Alpha-numeric text (maximum 200 characters). May be null.	The language in the file header.
ProductName	Alpha-numeric text (maximum 200 characters). May be null.	The product name in the file header.
ProductVersion	Alpha-numeric text (maximum 200 characters). May be null.	The product version number in the file header.

Inventory Object: ILMTPVUCounts

`ILMTPVUCounts` objects are uploaded to the `ImportedILMTPVUCounts` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

This table allows the 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.
`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.

Property	Attributes	Notes
SubCapacityPVU	Unsigned integer (int). Default: 0.	The number of sub-capacity PVU counts consumed for the license on the computer.
TitleName	Alpha-numeric text (maximum 512 characters). Mandatory. Database key.	The name of the title these points apply to.

Inventory Object: InstalledFileEvidence

InstalledFileEvidence objects are uploaded to the ImportedInstalledFileEvidence table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedInstalledFileEvidence table holds a record of the file evidence that has been installed on a computer from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ExternalFileID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the file evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
ExternalFilePathID	Unsigned integer (bigint). May be null.	The identifier used in the source connection for the path of the file evidence.

Property	Attributes	Notes
ExternalID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the computer that the file evidence is installed on. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: InstalledFileEvidenceUsage

InstalledFileEvidenceUsage objects are uploaded to the ImportedInstalledFileEvidenceUsage table in the operations (inventory) database.

The ImportedInstalledFileEvidenceUsage table holds a record of end-users that are using file evidence from the source connection.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ActiveTimeInSeconds	Unsigned integer (bigint). May be null.	The number of seconds that the file evidence was in use during the usage tracking period.
ExternalFileID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the file evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`ExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer that the file evidence is installed on. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalUserID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the end-user that has used the file evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`LastUsedDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The last used date of the file evidence.
`NumberOfSessions`	Unsigned integer (`bigint`). May be null.	The number of sessions that the file evidence was in use during the usage tracking period.
`StartDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The start date of the file evidence usage tracking period.

Inventory Object: InstalledInstallerEvidence

`InstalledInstallerEvidence` objects are uploaded to the `ImportedInstalledInstallerEvidence` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedInstalledInstallerEvidence` table holds a record of the installer evidence that has been installed on a computer from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`DiscoveryDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The date that the installer evidence was first seen.
`ExternalComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer that the installer evidence is installed on. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalInstaller EvidenceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
ExternalInstanceID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the instance that the installer evidence is associated with. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
InstallDate	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The install date of the installer evidence.

Inventory Object: InstalledInstallerEvidenceAttribute

InstalledInstallerEvidenceAttribute objects are uploaded to the ImportedInstalledInstallerEvidenceAttribute table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedInstalledInstallerEvidenceAttribute table holds a record of the values of the instance attributes for each installer evidence which is reported to be installed on a computer.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
AttributeID	Unsigned integer (int). Mandatory. Database key.	The identifier used in the source connection for the instance attribute.

Property	Attributes	Notes
`ExternalComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer that the installer evidence is installed on. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalInstaller EvidenceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalInstanceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the instance that the installer evidence is associated with. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`Value`	Alpha-numeric text (maximum 2Gb storage, or about 1 billion double-byte characters).	The value of the instance attribute for the installed installer evidence.

Inventory Object: InstalledInstallerEvidenceUsage

`InstalledInstallerEvidenceUsage` objects are uploaded to the `ImportedInstalledInstallerEvidenceUsage` table in the operations (inventory) database.

The `ImportedInstalledInstallerEvidenceUsage` table holds a record of installed evidence being used from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ExternalID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer that the installer evidence is installed on. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalInstallerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`ExternalInstanceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the instance that the installer evidence is associated with. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalUserID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the user that the installer evidence was used on. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`LastUsedDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The last used date of the installed installer evidence.
`NumberOfSessions`	Unsigned integer (`bigint`). May be null.	The number of sessions that the installer evidence was in use during the usage tracking period.
`StartDate`	An ASCII string of alphanumeric characters and punctuation (length 10 characters). May be null.	The start date of the installer evidence usage tracking period.

Inventory Object: InstalledWMIEvidence

`InstalledWMIEvidence` objects are uploaded to the `ImportedInstalledWMIEvidence` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedInstalledWMIEvidence` table holds a record of the WMI evidence that has been installed on a computer from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ExternalComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer that the WMI evidence is installed on. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ExternalEvidenceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the WMI evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
InstanceName	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The name of the WMI class instance used in the source connection for the WMI evidence ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: InstallerEvidence

InstallerEvidence objects are uploaded to the ImportedInstallerEvidence table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedInstallerEvidence table holds all of the installer evidence which has been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
AccessModeID	Unsigned integer (int). Default: 1. May be null.	The access mode ID of the file evidence.
DisplayName	Alpha-numeric text (maximum 256 characters). May be null.	The display name of the software as reported by the installer evidence.
Evidence	Alpha-numeric text (maximum 32 characters). May be null.	Identifier for the type of installer evidence.

Property	Attributes	Notes
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

InstallerEvidenceRepackageMapping objects are uploaded to the ImportedInstallerEvidenceRepackageMapping table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedInstallerEvidenceRepackageMapping table is used by the importer to map the original and current installer evidence of repackaged softwares as reported by the ISO tag evidence.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
CurrentDisplayName	Alpha-numeric text (maximum 256 characters). May be null.	The current display name of the repackaged software as reported by the ISO tag evidence.
CurrentPublisher	Alpha-numeric text (maximum 200 characters). May be null.	The current publisher of the repackaged software as reported by the ISO tag evidence.

Property	Attributes	Notes
`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 loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: Instance

`Instance` objects are uploaded to the `ImportedInstance` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedInstance` table holds all of the instances which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AuditEvidence`	Binary data from a file (maximum 2Gb). May be null.	Oracle LMS CVS files in zip archive.
`AuditEvidenceDate`	Date/time field. May be null.	Oracle LMS CSV files collection date.
`ExternalComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`InstanceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the instance. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`InstanceName`	Alpha-numeric text (maximum 256 characters). May be null.	The name of the instance.

Property	Attributes	Notes
ParentInstanceID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the parent instance. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: InstanceUser

InstanceUser objects are uploaded to the ImportedInstanceUser table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedInstanceUser table holds all of the end-users of an instance which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
AccountStatus	Alpha-numeric text (maximum 256 characters). May be null.	The current status of the end-user account.
ApplicationID	Alpha-numeric text (maximum 400 characters). Mandatory. Database key.	The Oracle EBS application ID the user has access to. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
ComputerID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the computer.
CreationDate	Date/time field. May be null.	The date and time when the end-user was created.

Property	Attributes	Notes
DefaultTablespace	Alpha-numeric text (maximum 256 characters). May be null.	The default tablespace for an Oracle end-user.
ExternalID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the instance end-user.
InstanceID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the instance.
LastLogonDate	Date/time field. May be null.	The date and time when the end-user last logged on to the computer.
TempTablespace	Alpha-numeric text (maximum 256 characters). May be null.	The temporary tablespace for an Oracle end-user.

Inventory Object: LicenseUser

LicenseUser objects are uploaded to the ImportedLicenseUser table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedLicenseUser table holds all of the external end-users (such as those used by Oracle or SAP licenses) retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
CostCenter	Alpha-numeric text (maximum 128 characters). May be null.	The cost center of the external end-user, as reported in SAP. Does not necessarily map to a cost centre in the GroupEx table.
Description	Alpha-numeric text (maximum 400 characters). May be null.	The description of the external end-user.
Email	Alpha-numeric text (maximum 400 characters). May be null.	An e-mail address for the external end-user.
EmployeeNumber	Alpha-numeric text (maximum 256 characters). May be null.	The employee number of the external end-user.

Property	Attributes	Notes
ExternalID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the external end-user. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
FirstName	Alpha-numeric text (maximum 256 characters). May be null.	The first name of the external end-user.
LastName	Alpha-numeric text (maximum 256 characters). May be null.	The last name or surname of the external end-user.
LicenseUserID	Unsigned integer (int). May be null.	The identifier of the external end-user in the LicenseUser table that this imported end-user links to. This is populated by the import process and does not need to be provided by the source connections.
UserName	Alpha-numeric text (maximum 400 characters). May be null.	The name of the external end-user.

Inventory Object: RelatedInstalledInstallerEvidence

RelatedInstalledInstallerEvidence objects are uploaded to the ImportedRelatedInstalledInstallerEvidence table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedRelatedInstalledInstallerEvidence table holds parent-child relationship between installer evidence.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`ChildExternal ComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer that the installer evidence is installed on. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ChildExternal InstallerEvidenceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ConfidenceLevel`	Unsigned integer (`int`). May be null.	Confidence level for each bundled installer evidence (as a percentage).
`IsCharged`	Boolean (0 or 1). Mandatory. Database key.	The identifier used in the source connection to determine the pricing relation between parent and child installer evidence (specifies if it is charged = 1 or free = 0). ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
`ParentExternal ComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the computer that the installer evidence is installed on. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`ParentExternal InstallerEvidenceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: RemoteUserToApplicationAccess

`RemoteUserToApplicationAccess` objects are uploaded to the `ImportedRemoteUserToApplicationAccess` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedRemoteUserToApplicationAccess` table stores the applications that remote users have access to

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
AccessModeID	Unsigned integer (int). Mandatory. Database key.	The access mode ID for the remote application access. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
ExternalFileID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the file evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
ExternalInstaller EvidenceID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the installer evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
ExternalServerID	Unsigned integer (bigint). Mandatory. Database key.	The External Server ID for the remote server. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
ExternalUserID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the end-user that has used the file evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
LastUsedDate	Date/time field. May be null.	The last time the remote application was used by the user.
VDIGroupUUID	A universally unique identifier. May be null.	The desktop group UUID from which the application is published

Inventory Object: Site

Site objects are uploaded to the ImportedSite table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedSubnet contains sites imported from Microsoft Active Directory

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
AutoPopulated	Boolean (0 or 1). Default: 0.	Is the site auto populated at source?
Enabled	Boolean (0 or 1). Default: 1.	Is the site enabled?

Property	Attributes	Notes
Name	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The site's name.

Inventory Object: SiteSubnet

SiteSubnet objects are uploaded to the ImportedSiteSubnet table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedSiteSubnet contains sites and subnets imported from Microsoft Active Directory

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
AutoPopulated	Boolean (0 or 1). Default: 0.	Is the subnet auto populated at source?
Enabled	Boolean (0 or 1). Default: 1.	Is the subnet enabled?
IPSubnet	Alpha-numeric text (maximum 64 characters). Mandatory. Database key.	The IP subnet.
IPSubnetBits	UNRECOGNIZED TYPEMandatory. Database key.	The IP subnet mask in CIDR notation.
SiteName	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The site's name.

Inventory Object: SoftwareLicense

SoftwareLicense objects are uploaded to the ImportedSoftwareLicense table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedSoftwareLicense table holds all of the licenses which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
EntitlementCount	Unsigned integer (int). May be null.	The number of entitlements for the license.
ExpiryDate	Date/time field. May be null.	The expiry date of a subscription license.

Property	Attributes	Notes
ExternalLicenseID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the license. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
IsSubscription	Boolean (0 or 1). Default: 0.	Indicates whether or not the license is a subscription license.
LicenseName	Alpha-numeric text (maximum 256 characters). May be null.	The name of the license.
PartNo	Alpha-numeric text (maximum 100 characters). May be null.	The publisher's part number for this license.
SoftwareLicenseID	Unsigned integer (int). May be null.	Identifier of the license in the SoftwareLicense table that this imported license links to. This is populated by the import process and does not need to be provided by the source connections.
SoftwareLicenseTypeID	Unsigned integer (int). May be null.	The license type ID of the license.

Inventory Object: SoftwareLicenseAllocation

SoftwareLicenseAllocation objects are uploaded to the ImportedSoftwareLicenseAllocation table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedSoftwareLicenseAllocation table holds the links between licenses and end-users which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ExternalLicenseID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the license. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
ExternalUserID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the license. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: User

User objects are uploaded to the ImportedUser table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedUser table holds all of the end-users which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ComplianceDomainID	Unsigned integer (int). May be null.	Identifier of the domain in the ComplianceDomain table that this end-user belongs to. This is populated by the import process and does not need to be provided by the source connections.

Property	Attributes	Notes
ComplianceUserID	Unsigned integer (int). May be null.	Identifier of the end-user in the ComplianceUser table that this imported user links to. This is populated by the import process and does not need to be provided by the source connections.
CostCenter	Alpha-numeric text (maximum 128 characters). May be null.	The cost center of the end-user, as reported in SAP. Does not necessarily map to a cost centre in the GroupEx table.
Domain	Alpha-numeric text (maximum 100 characters). May be null.	The domain of the end-user.
Email	Alpha-numeric text (maximum 200 characters). May be null.	The email address of the end-user.
EmployeeNumber	Alpha-numeric text (maximum 128 characters). May be null.	The employee number of the end-user.
ExternalID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the end-user. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
FirstName	Alpha-numeric text (maximum 128 characters). May be null.	The first name of the end-user.
InventoryAgent	Alpha-numeric text (maximum 64 characters). May be null.	The name of the person or tool that performed the last inventory. For imported spreadsheets, you may wish to include the name of the person preparing the data, in case there is subsequent follow-up required.

Property	Attributes	Notes
IsBlacklisted	Boolean (0 or 1). Default: 0.	This is populated by the import process and does not need to be provided by the source connections. The field is set to True if the end-user matches a record from the UserNameBlacklist table, meaning the account should not be included in compliance calculations.
LastName	Alpha-numeric text (maximum 128 characters). May be null.	The last name or surname of the end-user.
MapUsingEmailAddress	Boolean (0 or 1). Default: 0.	Indicates whether or not the user's email address should be used to try and map it to an existing ComplianceUser record.
SAMAccountName	Alpha-numeric text (maximum 64 characters). May be null.	The SAM account name of the end-user.
UserName	Alpha-numeric text (maximum 64 characters). May be null.	The account name of the end-user.

Inventory Object: VDI

VDI objects are uploaded to the ImportedVDI table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedVDIUser table stores the list of VDI devices, their master VM template and the VDI group the VDI device resides under.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ApplicationDelivery Only	Boolean (0 or 1). May be null.	Determines whether the VDI device is used only to server applications.

Property	Attributes	Notes
BrokerType	Alpha-numeric text (maximum 64 characters). Mandatory. Database key.	The broker type of the VDI device. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
ComputerName	Alpha-numeric text (maximum 64 characters). May be null.	The computer name of the VDI.
Domain	Alpha-numeric text (maximum 100 characters). May be null.	The domain name of the VDI device.
ExternalDeviceID	Unsigned integer (bigint). May be null.	The identifier used in the source connection for the VDI device.
IsPersistent	Boolean (0 or 1). May be null.	Determine whether the VDI device is a persistent VDI device.
SiteName	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The site name of the VDI. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
TemplateName	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	The VDI template the VDI is cloned from. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
VDIGroupName	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	The VDI group the VDI device belongs to. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
VDIGroupUUID	A universally unique identifier. May be null.	The group UUID the VDI device belongs to.

Inventory Object: VDITemplate

VDITemplate objects are uploaded to the ImportedVDITemplate table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedVDITemplate table stores the list of VDI templates.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`BrokerType`	Alpha-numeric text (maximum 64 characters). Mandatory. Database key.	The broker type of the VDI template. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`SiteName`	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The site name of the VDI. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`TemplateName`	Alpha-numeric text (maximum 64 characters). Mandatory. Database key.	The template name of the VDI template. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`VDITemplateExternalID`	Unsigned integer (`bigint`). May be null.	The ExternalID of the VDI template in the `ImportedComputer` table.

Inventory Object: VDIUser

`VDIUser` objects are uploaded to the `ImportedVDIUser` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedVDIUser` table stores the list of users that have been granted access to VDI groups.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`BrokerType`	Alpha-numeric text (maximum 64 characters). May be null.	The broker type of the VDI for the end user.
`ExternalUserID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the end-user that has access to the VDI. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`SiteName`	Alpha-numeric text (maximum 256 characters). May be null.	The site name of the VDI.
`VDIGroupName`	Alpha-numeric text (maximum 100 characters). May be null.	The VDI group the end-user has access to.

Inventory Object: VMHostManagedBySoftware

`VMHostManagedBySoftware` objects are uploaded to the `ImportedVMHostManagedBySoftware` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedVMHostManagedBySoftware` table contains relationships between installer evidence of management software and VM hosts it manages.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ExternalComputerID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the computer that the management software installer evidence is installed on.
ExternalInstallerID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for an installer evidence of management software.
ExternalVMHostID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the VM host computer that is managed by a management software.
RelationType	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	Identifier for the type of relation, to be matched against ImporterString column of RelationType table.

Inventory Object: VMPool

VMPool objects are uploaded to the ImportedVMPool table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedVMPool table holds all of the virtual machine pools which have been retrieved from the source connections and the number of processors and cores that are assigned to each pool.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
HostComputerID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the computer which is hosting the pool. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
NumberOfCores	Fractional number (float). May be null.	The number of cores available to this pool.
NumberOfProcessors	Fractional number (float). May be null.	The number of processors available to this pool.

Property	Attributes	Notes
ObjectType	Alpha-numeric text (maximum 256 characters). Mandatory. Database key.	The type of pool. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
ParentName	Alpha-numeric text (maximum 100 characters). May be null.	The name of the parent pool. This is the PoolName property for the parent pool.
ParentObjectType	Alpha-numeric text (maximum 256 characters). May be null.	The type of pool of the parent.
PoolFriendlyName	Alpha-numeric text (maximum 256 characters). May be null.	The friendly name of the pool.
PoolName	Alpha-numeric text (maximum 100 characters). Mandatory. Database key.	The name of the pool. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Inventory Object: VirtualMachine

`VirtualMachine` objects are uploaded to the `ImportedVirtualMachine` table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The `ImportedVirtualMachine` table holds all of the virtual machines which have been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
`AffinityEnabled`	Boolean (0 or 1). Default: 0.	Set this to `True` if this VM is unable to move to different host computers.
`CPUAffinity`	Alpha-numeric text (maximum 256 characters). May be null.	Contains the CPU Affinity value for virtual machine(Host Logical Processors)
`CPUUsage`	Unsigned integer (`int`). May be null.	The maximum CPU usage of the virtual machine (MHz).
`ComputerName`	Alpha-numeric text (maximum 256 characters). May be null.	The computer name of the virtual machine.
`CoreAffinity`	Alpha-numeric text (maximum 256 characters). May be null.	Contains the Core Affinity value for virtual machine
`FriendlyName`	Alpha-numeric text (maximum 256 characters). May be null.	The friendly name of the virtual machine.
`GuestFullName`	Alpha-numeric text (maximum 256 characters). May be null.	Configured operating system for the guest.
`HostComputerID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the virtual machine's host computer. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*

Property	Attributes	Notes
InventoryAgent	Alpha-numeric text (maximum 64 characters). May be null.	The name of the person or tool that performed the last inventory.
Manufacturer	Alpha-numeric text (maximum 128 characters). May be null.	The manufacturer of the virtual machine.
MemoryUsage	Unsigned integer (bigint). May be null.	The maximum memory usage of the virtual machine (bytes).
ModelNo	Alpha-numeric text (maximum 128 characters). May be null.	The model number of the virtual machine.
NumberOfHardDrives	Unsigned integer (int). May be null.	The number of hard drives in the virtual machine.
NumberOfNetworkCards	Unsigned integer (int). May be null.	The number of network cards in the virtual machine.
NumberOfProcessors	Unsigned integer (int). May be null.	The number of processors in the virtual machine.
PartitionID	Alpha-numeric text (maximum 100 characters). May be null.	Partition ID generated and used by the managing virtualization platform
PartitionNumber	Unsigned integer (int). May be null.	Number of this partition
PoolName	Alpha-numeric text (maximum 100 characters). May be null.	The name of the pool that the virtual machine belongs to.
PoolType	An ASCII string of alphanumeric characters and punctuation (length 100 characters). May be null.	The type of the pool that the virtual machine belongs to.
ProcessorType	Alpha-numeric text (maximum 256 characters). May be null.	The type of processor in the virtual machine.
TotalMemory	Unsigned integer (bigint). May be null.	The total RAM in the computer, in bytes.
UUID	Alpha-numeric text (maximum 256 characters). May be null.	The UUID of the virtual machine.

Property	Attributes	Notes
VMComputerID	Unsigned integer (bigint). Mandatory. Database key.	The identifier used in the source connection for the virtual machine's computer. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
VMEnabledStateID	Unsigned integer (int). May be null.	The state of the machine (powered on, off, etc).
VMLocation	Alpha-numeric text (maximum 256 characters). May be null.	Location of the virtual machine on the file system.
VMName	Alpha-numeric text (maximum 256 characters). May be null.	The name of the virtual machine.
VirtualMachineType	An ASCII string of alphanumeric characters and punctuation (length 100 characters). May be null.	The type of virtual machine.

Inventory Object: WMIEvidence

WMIEvidence objects are uploaded to the ImportedWMIEvidence table in the operations (inventory) database. Multiple imports will merge updated data with existing records, and add new records as applicable.

The ImportedWMIEvidence table holds all of the WMI evidence which has been retrieved from the source connections.

Attributes are listed here in alphabetical order.

Property	Attributes	Notes
ClassName	Alpha-numeric text (maximum 50 characters). May be null.	The WMI class name of the WMI evidence.

Property	Attributes	Notes
`ExternalEvidenceID`	Unsigned integer (`bigint`). Mandatory. Database key.	The identifier used in the source connection for the WMI evidence. ***Note:*** *Strictly, this attribute may be null, because it forms part of a compound database key. However, null values may cause import errors (where this object does not get imported), and multiple records from the same connection having nulls may cause data loss through over-writing. It is therefore best practice to treat this field as mandatory.*
`PropertyName`	Alpha-numeric text (maximum 50 characters). May be null.	The WMI property name of the WMI evidence.
`PropertyValue`	Alpha-numeric text (maximum 256 characters). May be null.	The value of the property of the WMI evidence.

8

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 you 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?

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
- Immediately uploads the package to the operations database
- Repeats this process for each of the other currently enabled adapters awaiting execution.

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
- Move the complete adapter into production.

Separately, 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.

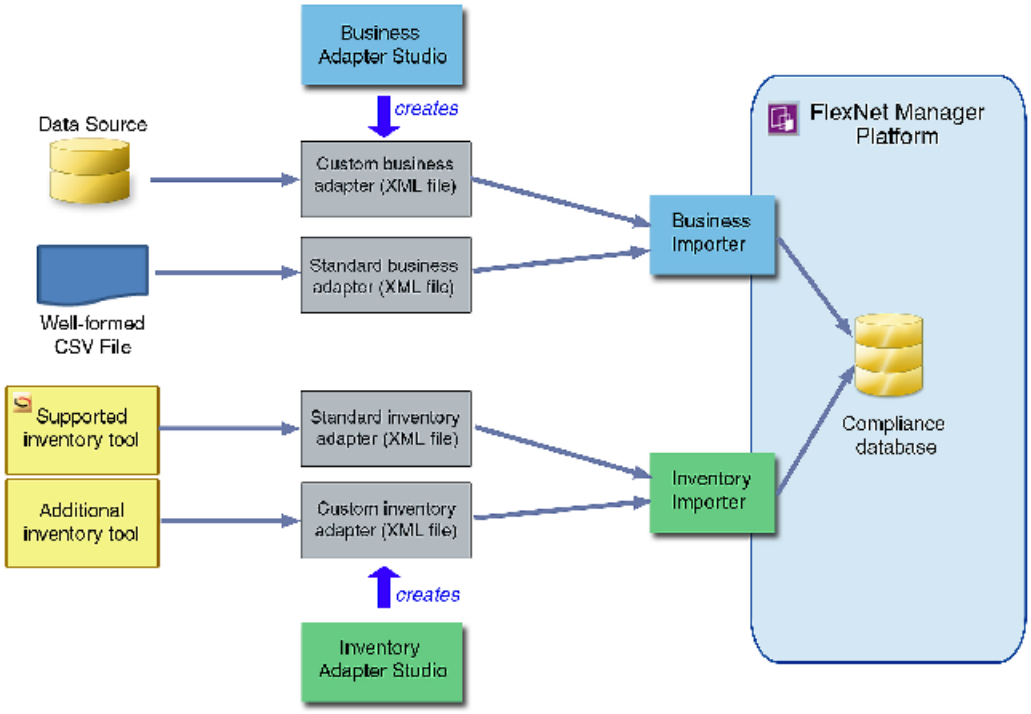
Prerequisites

In brief:

- **System requirements:** Included in the requirements for the inventory beacon. 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.
- **Installation:** The Business Adapter Studio 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.
- **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, as the Business Adapter Studio running on the inventory beacon does not allow for any custom SQL.

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.


Business Adapter Studio
creates
Data Source
Custom business adapter (XML file)
Standard business adapter (XML file)
Well-formed CSV File
FlexNet Manager Platform
Business Importer
Compliance database
Supported inventory tool
Additional inventory tool
Standard inventory adapter (XML file)
Custom inventory adapter (XML file)
Inventory Importer
creates
Inventory 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.

You build, review, and run your business adapter on your inventory beacon.

To develop a business adapter (overview):

1. Launch Business Adapter Studio and add a framework for a new adapter (see To Start the Business Adapter Studio).
2. Configure the connection to the data source. This is where data will be imported from. See Connecting to a Data Source.
3. Confirm that you are querying the correct data from the data source. See Reviewing Data from the Source.
4. 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.
5. 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.
6. Define rules that manage updates and creation of these objects, based on the incoming data. See Defining Import Rules for a Database Item.
7. 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.
8. Save the adapter (Saving Business Adapters).

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.

To Start the Business Adapter Studio

Start the Business Adapter Studio from your inventory beacon interface.

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.`

To start the Business Adapter Studio:

1. On the inventory beacon:
   a. Select the **Business Importer** tab.
   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.

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.)

Choose either of these options when the Business Adapter Studio is already open:

- 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.

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.

As the adapter name will be referenced by the business importer, you should ensure it has a useful name that will assist future maintenance.

To rename a business adapter:

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, 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**.

To change the XML file name:

- From the **File** menu, choose **Save As**.

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).

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*.

The collected information is zipped into an archived package on the inventory beacon, and automatically uploaded to the central application server. There, once a night, all uploaded packages are imported into the operations databases. You do not need to specify any connection details for the upstream target (or destination) database.

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.

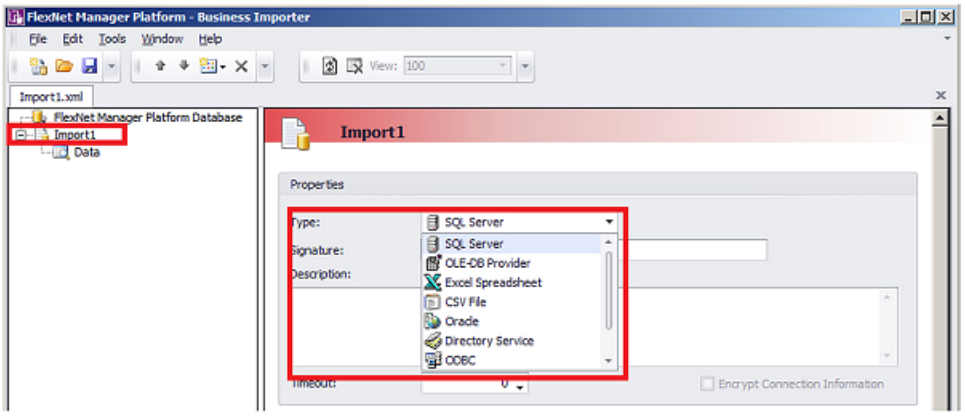
All the data source adapters start with these common details. Next you will select a different help page for details on completing this task, depending on the type of data source you are working with.

To initialize a data connection:

1. Ensure that the desired adapter is open in Business Adapter Studio.
2. In the structure tree on the left, select the topmost node identifying your business adapter.

   A page of properties for your business adapter is displayed.
3. From the **Type** drop-down list, select one of the available options. Your choice changes the lower part the page, and each option is documented below.

Figure 6: 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:

`SQL Server`	Connect with a Microsoft SQL Server database (see Completing Connection Properties for Database Sources)
`OLE-DB Provider`	Use for any data source that provides an OLE-DB compliant interface (see CCompleting Connection Properties for Database Sources)
`Excel Spreadsheet`	See Completing Connection Properties for Excel Spreadsheets.

`CSV File`	Use for a file of comma-separated values (see Completing Connection Properties for CSV Files) ***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)
`Directory Service`	Use for an LDAP directory service such as Microsoft Active Directory (see Completing Connection Properties for Directory Services)
`ODBC`	Use this for a data source that provides an ODBC compliant interface (see Completing Connection Properties for Database Sources)
`Web Service`	Use for a SOAP web service (see Completing Connection Properties for Web Services)
`XML`	Use for an XML file (see Completing Connection for XML Files).

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.

To complete connection properties for these database sources:

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** drop-down 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.

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:
	`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.
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 inventory beacon where the business adapter runs.

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.
`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. ***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 License Statement (MLS).

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.

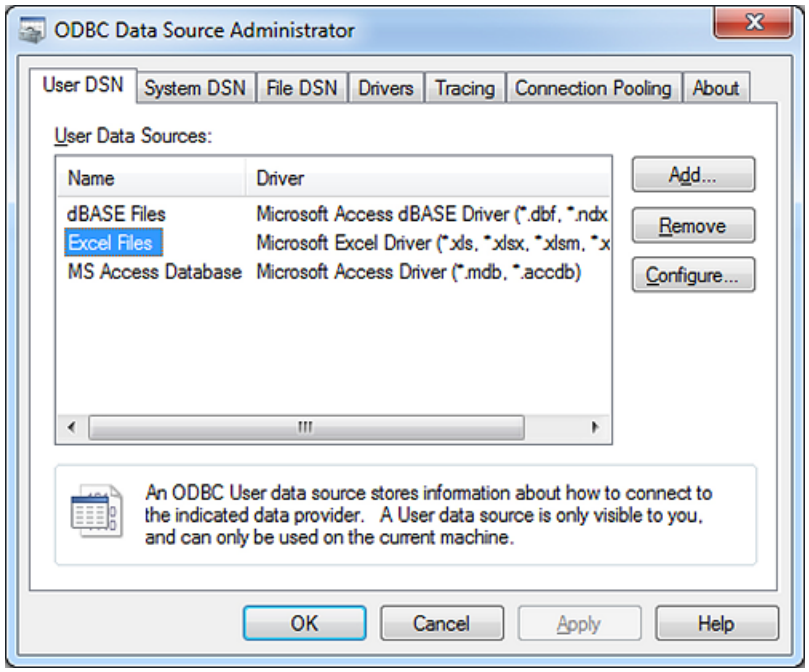
To generate and adjust the `schema.ini` file:

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 7: The ODBC Data Source Administrator screen

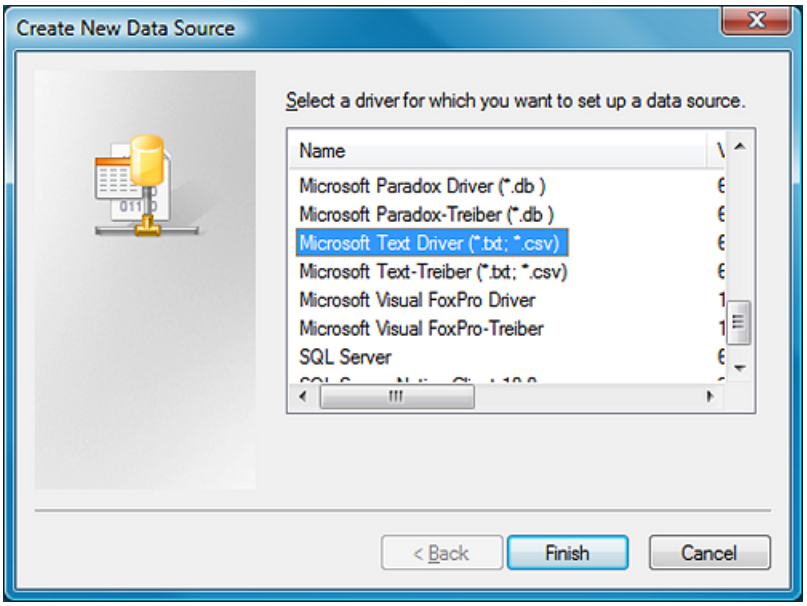


   Note: *This tool is primarily used to create and manage ODBC data sources. However, it is used here simply to create a default `schema.ini` file.*

2. Click **Add...**.

   The **Create New Data Source** dialog is displayed.

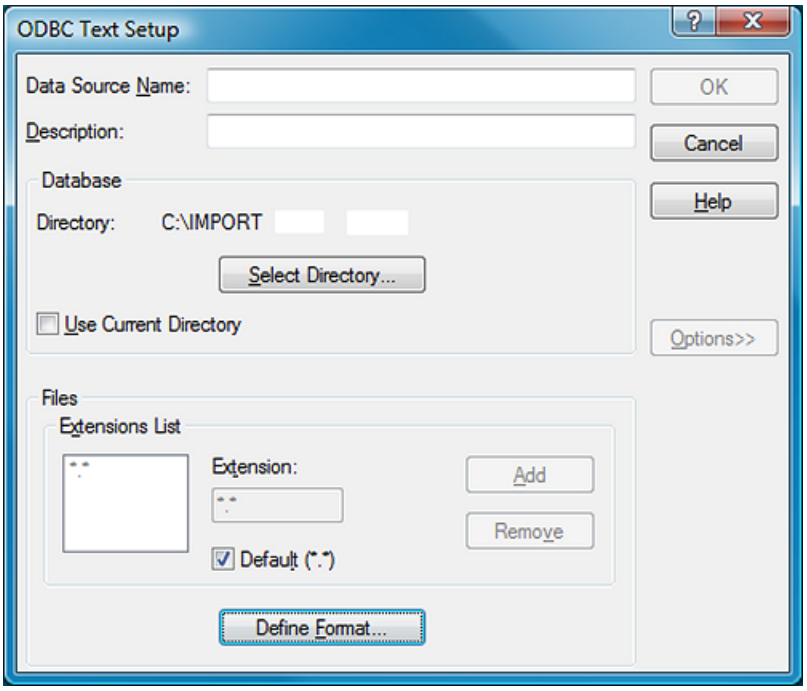
Figure 8: Pick the driver matching your data source



3. For CSV files or text files, select the **Microsoft Text Driver**.
4. Click **Finish**.

   The **ODBC Text Setup** dialog is displayed.

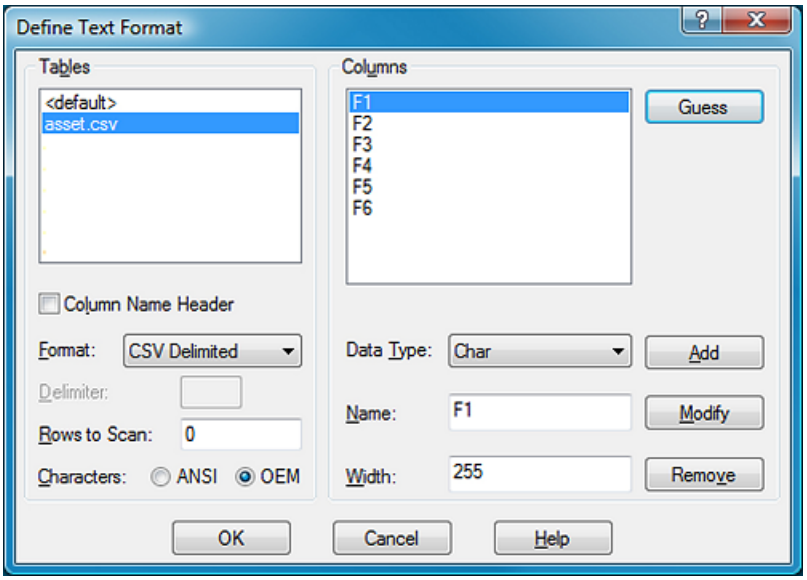
Figure 9: Locate the CSV (or text) file



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 10: 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

You can import a number of properties from your directory service, most commonly from Microsoft Active Directory.

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.*

Property	Notes
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 • 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.

Property	Notes
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.

Properties	Notes

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): • 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.

Properties	Notes

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.*

Reviewing Data from the Source

You can validate the returned data, with no effect on the source data or future imports.

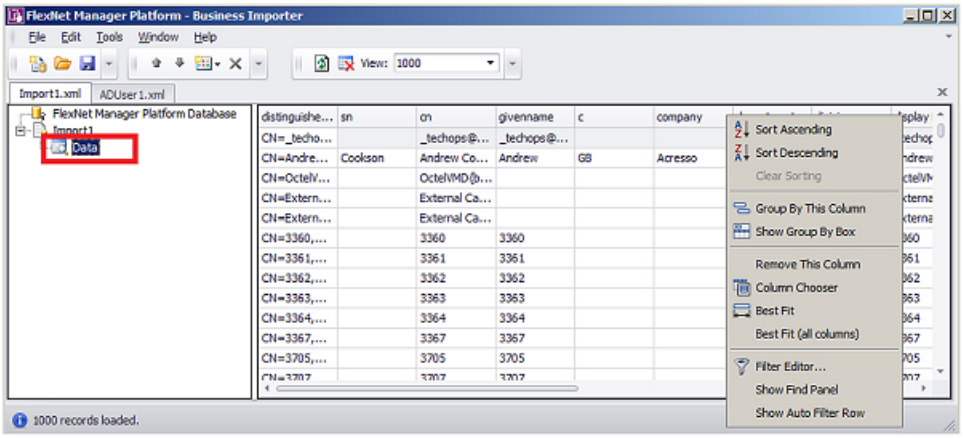
Once the data source connection has been configured, the next step is to preview the data returned from the query. This confirms that you have configured the connection properly and are retrieving the expected data.

To review the returned 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).
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).
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 for database objects, and Defining Import Rules for Attributes/Properties 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. For their properties, see Defining Import Rules for Attributes/Properties.

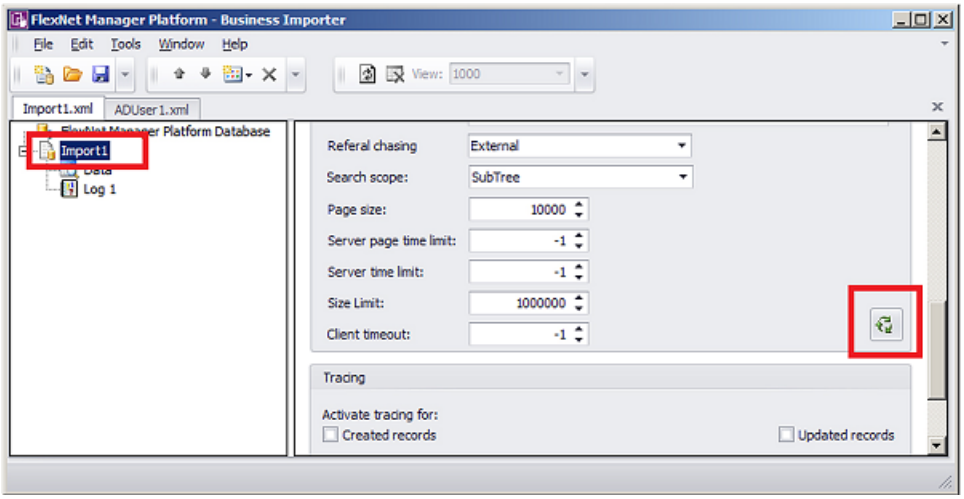
Retrieving the List of Fields

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.

To retrieve the list of fields:

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 Adapter 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 inventory 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.

To manually update the data model and adapter templates:

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:*

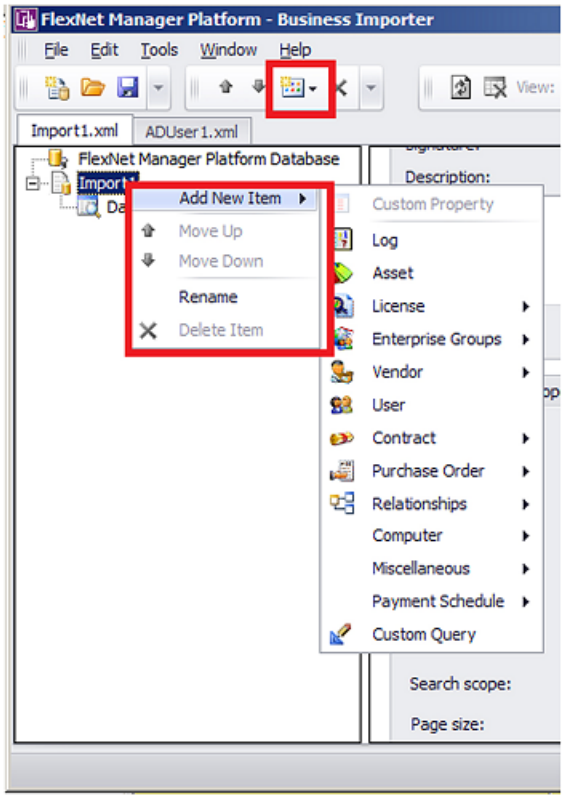
- *The custom property has been defined on the application server*
- *You have downloaded the latest templates and data schema that includes your custom properties. For details, see Updating Business Adapter Templates and Data Model.*

To select database items:

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
- 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.

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.

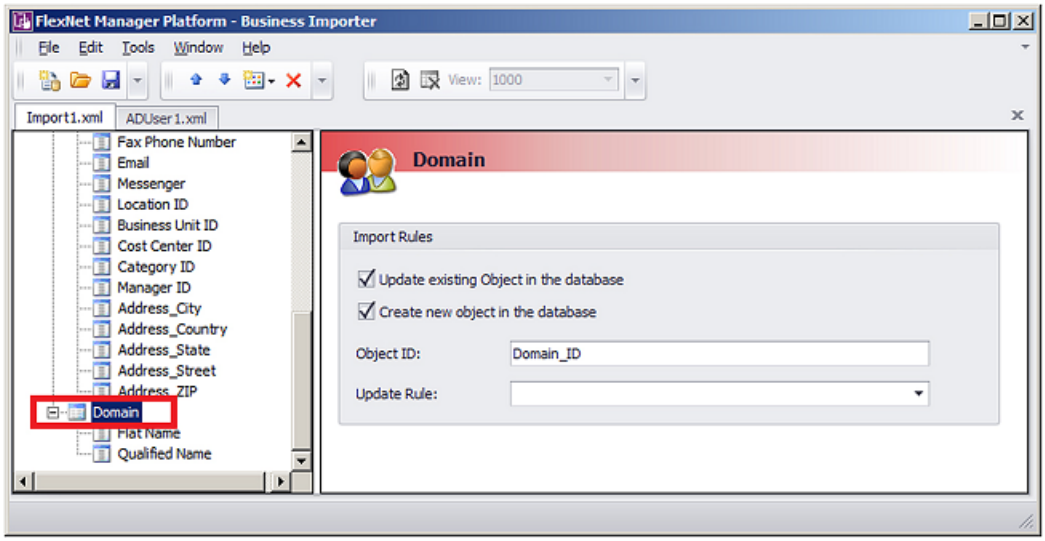
Your business adapter is open in Business Adapter Studio.

To specify import rules for database objects:

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`).

Option	Description
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. These choices control how the Business Importer should handle relationships already existing in the database that are not supported by evidence included in the current import through this adapter. • **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 ◦ `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

Option	Description

- `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
  - `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

Option	Description

- `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** (covers individual allocations made on licenses that may influence consumption calculated for linked software applications)
  - `Add new links, leave the existing ones untouched` — (default) Never delete any allocations already existing in the database
  - `Detach unfound software allocations from the considered computers` — Removes license allocation links from computers included in the import, where application installations linked to the same license were not found in the incoming data
  - `Detach unfound computers from considered software allocations` — Removes license allocations (mentioned in the import) from those previously-linked computers that were not also found individually listed in the incoming data
  - `Detach all unfound links between the considered computers and software allocations` — All computers and license allocations mentioned in the imports have their existing records in the database checked; and any links in the database that are not also repeated in the incoming data are removed from the database.
  - `Reject duplicate records` — Duplicate records of allocations are recorded in the log file, rather than being silently ignored.

Defining Import Rules for Attributes/Properties

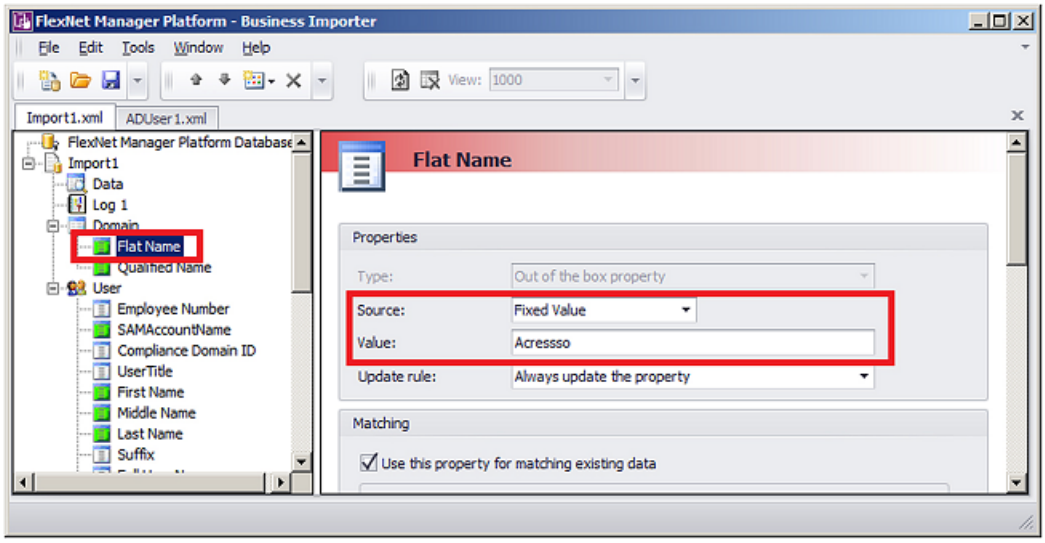
When you have set the import rules for an item in the compliance database/>, you may set fine-grain import rules for handling imports to the individual properties of each of those objects.

To define rules for property imports:

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:

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.
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.
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.

If null value is found	Select one of the following values from the drop-down 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 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	This setting is permanently disabled on your inventory beacon.
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 **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.

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 sign (`#`). 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.` • **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 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.

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: • `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 drop-down 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. When entering an SQL statement, the following keywords can be used: • `[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.

Testing and Diagnosis for Your Business Adapter

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.*

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.)
Monetary values appear without any currency units	The rate ID for this value is missing from your imported data.

9

FlexNet Report Designer

FlexNet Report Designer, powered by Cognos, allows you to build custom reports based on weekly snapshots of your licensing and installation data.

More About FlexNet Report Designer

Data for reports is automatically copied from the compliance database to the data warehouse database once a week at 6am (central server time) on Sunday mornings.

The system preserves the 12 most recent weekly snapshots. As well, the last snapshot taken within a month is preserved as a monthly snapshot, and the 36 most recent monthly snapshots are preserved. (The other weekly snapshots taken earlier each month are automatically culled as they are outside the 12 most recent.)

Tip: *Even though FlexNet Report Designer was only made available in cloud at the 2016 R1 release of FlexNet Manager Suite, snapshots have been collected on the above schedule for about the previous two years, or since you started using the cloud implementation (whichever is the most recent). This historical data is now available for your use in reports.*

The cloud implementation of FlexNet Report Designer has the following limitations:

- The `Analytics Administrator` role is not available.
- Cognos log files are not available.
- You cannot include any custom SQL in reports. If you attempt to do so, you will see an error `You do not have permission to create or edit SQL/MDX.`

If the default number of operators in each Cognos role is not sufficient for your needs, please contact your Flexera Software Consultant to request additional quantities at no additional cost.

Before attempting to access FlexNet Report Designer, make sure that you have added the operator accounts to the appropriate role:

1. Navigate to the system menu (▼ in the top right corner), and choose **Accounts**.
2. In the **Roles** tab, expand the **Business reporting portal** section.
3. Select the appropriate privileges from the drop-down list (such as `Analytics User`).

4. Give the role an appropriate name and description, and click **Create**.
5. Switch to the **All Accounts** tab, and create or select your account, and assign it to the new role.

Data Models for FlexNet Report Designer

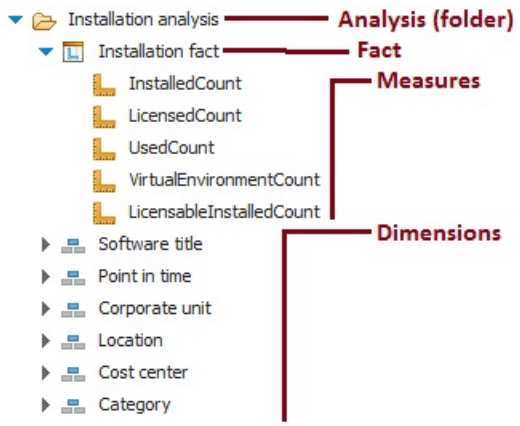
A single data model package is available in the cloud implementation of FlexNet Manager Suite for use in FlexNet Report Designer:

- **FlexNet Manager Platform Data Warehouse (analysis)** contains a dimensional model that allows considerable flexibility for online analytical processing (OLAP). This data comes from the data warehouse database (suggested name: FNMSDataWarehouse), and some report-time specialized queries that calculate trending data sets and the like.

The model consists of the following elements:

- *Folder* — In the dimensional model, each folder is called an *analysis*, including **Installation analysis** for reports about software inventory within your enterprise; and **Consumption analysis** for reports about license positions.
- *Fact* — In the dimensional model, the first child of each analysis folder is the central fact around which the related data dimensions and measures are organized. For example, the **Installation analysis** contains the **Installation fact**; and similarly, the **Consumption analysis** starts with the **Consumption fact**.
- *Dimension* — In the dimensional model, each dimension gives a related data set that you can use to analyze the central fact. For example, it's obvious that when you report on your **Installation fact**, you need to analyze in terms of individual software applications; so that there is a **Software title** dimension available.
- *Measure* — Each measure contains a specific value derived (in general) from one or more fields in one of the source databases. In the dimensional model, some measures are attached directly to the fact. For example, the **Installation fact** has an **Installed (max qty)** measure that you can include in your reports. Measures also relate to other dimensions, but here they are referred to as *attributes*, and you generally choose from individual *values* of the attribute, rather than selecting the entire attribute. For example, the **Software title** dimension expands into a hierarchy of the **Publisher**, **Product**, and **Application** measures; but for creating reports, you generally select one of the publisher's names, and so on.

Figure 17: Elements in the dimensional data model



The dimensional data model is covered in the following sections.

Data Warehouse (Analysis) Model

FlexNet Manager Platform Data Warehouse (analysis) dimensional model centers around two related facts for analysis:

- **Installation fact**
- **Consumption fact**.

Because these facts are so closely related, they share several dimensions in common. These shared dimensions are grouped separately in this document, and described only once.

Installation Analysis

The **Installation analysis** is a folder which contains:

- The **Installation fact** (see Installation Fact: Measures)
- A **Software title** dimension that is unique to the **Installation analysis** (see Software Title Dimension)
- Several common dimensions shared with the **Consumption analysis** (see Point in Time Dimension and Enterprise Group Dimensions).

Together these dimensions and associated measures/attributes allow you to analyze software installation numbers over time in your enterprise.

Installation Fact: Measures

The **Installation fact** allows tracking of software installations over time. The **Installation fact** supports the following measures. These general comments apply to all measures:

- For each measure, the figure shown is the maximum value found in all data snapshots included in your reporting period.
- All values are filtered by the access rights of the current operator (or report user). Access rights are managed through roles to which the operator is assigned, using enterprise groups of various kinds as filters. For example, if operator Sam is denied access to data from your North American location (and its children), then a report entry for license consumption in your Chicago office always shows zero when viewed by Sam, regardless of how many licenses are really consumed in Chicago.
- Since you may build a report using any combination of enterprise groups, the totals described for the following measures may be segmented across those groups, as expected. This subtotalling and segmentation effect is omitted from the following descriptions for simplicity.

Table 8: Measures for the Installation fact (alphabetical listing)

Measure	Notes
Installed (max qty)	The total number of application installations, as filtered by your chosen dimensions and identified in the most recent compliance calculation before the snapshot. *Related management view:* **License Compliance > Installed Applications**.
Installed on VMs (max qty)	The subset of the total application installations that are on devices identified as virtual machines. This is not relevant to license consumption, since these installations may be covered by special rights, or by exemptions, or in other ways so that they do not consume from a license. *Related management view:* No direct equivalent.
Licensable installations (max qty)	The subset of the total application installations that appear to be licensable (regardless of whether or not they are currently consuming from a license). *Related management view:* In application properties, on the **Devices** tab, you can add the **Licensable** column, and filter for `Yes`. The results returned gives the equivalent count for the chosen application.
Licensed (max qty)	The subset of the total application installations that have been covered by a license (and are consuming license entitlements) in the most recent compliance calculation before the snapshot. *Related management view:* **License Compliance > Installed Applications**, filtered by application name and **Licensed** = `Yes`.
Used (max qty)	The subset of the total application installations that appear to be in use as at the most recent compliance calculation before the snapshot. *Related management view:* **License Compliance > Installed Applications**, filtered by application name and with **Usage** column displayed (this gives the count).

Software Title Dimension

As part of the **Installation analysis**, the **Software title** dimension allows you to drill down to an individual application (or 'software title') as part of analyzing software installed throughout your enterprise.

The **Software title** dimension expands into a three-level hierarchy that gives you access to (in order):

- **Publisher**
- **Product**
- **Software Title Name** (or, as seen within FlexNet Manager Suite, the application name).

You may report on any level. For example, if you choose only a publisher, your report includes all products and applications from that publisher.

Table 9: Attributes for the Software title dimension, Publisher level

Attribute	Notes
Publisher	The name of the company that publishes this application (and product). *Related management view:* For each application, the publisher is linked through the **General** tab of the application properties. However, the publisher must exist first for linking, and these records are maintained in the **Procurement > All Vendors** view. The publisher for each application is widely available in application listings, such as **License Compliance > All Applications**.

Table 10: Attributes for the Software title dimension, Product level

Attribute	Notes
Product	The common name for a family of applications, independent of version or edition details. This must be unique for any given publisher. *Related management view:* For each application, the **Product** name is displayed in the **General** tab of application properties (and when you manually create an application record, you can link the product value there). All applications that share a common value in that **Product** field are deemed to be distinct versions/editions of the same product. The product name is widely available in application listings, such as **License Compliance > All Applications** (although sometimes you must add it to the listing from the column chooser).

Table 11: Attributes for the Software title dimension, Software Title (or application) level (alphabetical listing)

Attributes	Notes
Application	The name of the application. *Related management view:* For each application, the **Name** is displayed in the **General** tab of application properties (and when you manually create an application record, you can edit the application name there). The application name appears in all application listings, such as **License Compliance > All Applications**.
Application status	Shows the current status for the application, from values such as `Authorized`, `Ignored`, `Deferred`. and others. *Related management view:* For each application, the **Status** is displayed in the **General** tab of application properties (and when you manually create an application record, you can edit the status there). The application **Status** is widely available in application listings, such as **License Compliance > All Applications**.

Attributes	Notes
Classification	The classification applied to this application, such as `Commercial` or `Malware`. *Related management view:* For each application, the **Classification** is displayed in the **General** tab of application properties (and when you manually create an application record, you can edit the classification there). The application classification is widely available in application listings, such as **License Compliance > All Applications**.
Edition	The edition of this application. Editions describe different levels of product functionality, such as Standard and Pro. *Related management view:* For each application, the **Edition** is displayed in the **General** tab of application properties (and when you manually create an application record, you can edit the edition there). The application edition is widely available in application listings, such as **License Compliance > All Applications**.
Is licensed	A Boolean that indicates whether the application is linked to any license. *Related management view:* For each application, the **Licenses** tab of application properties lists the license(s) to which this application is linked. You can also attach or detach licenses to/from the application on that tab. The **Licensed** column is widely available in application listings, such as **License Compliance > All Applications**.
Version	The version (or release number) of this application. *Related management view:* For each application, the **Version** is displayed in the **General** tab of application properties (and when you manually create an application record, you can edit the version there). The application version is widely available in application listings, such as **License Compliance > All Applications**.

Consumption Analysis

The **Consumption analysis** is a folder which contains:

- The **Consumption fact** (see Consumption Fact: Measures)
- A **Software license** dimension that is unique to the **Consumption analysis** (see Software License Dimension)
- Several common dimensions shared with the **Installation analysis** (see Point in Time Dimension and Enterprise Group Dimensions).

Together these dimensions and associated measures/attributes allow you to analyze license consumption numbers over time in your enterprise.

Consumption Fact: Measures

The **Consumption fact** allows tracking the results of license consumption calculations over time. It supports the measures listed below. These general comments apply to all measures:

- For each measure, the figure shown is the maximum value found in all data snapshots included in your reporting period.
- All values are filtered by the access rights of the current operator (or report user). Access rights are managed through roles to which the operator is assigned, using enterprise groups of various kinds as filters. For example, if operator Sam is denied access to data from your North American location (and its children), then a report entry for license consumption in your Chicago office always shows zero when viewed by Sam, regardless of how many licenses are really consumed in Chicago.
- Since you may build a report using any combination of enterprise groups, the totals described for the following measures may be segmented across those groups, as expected. This subtotalling and segmentation effect is omitted from the following descriptions for simplicity.

Table 12: Measures for the Consumption fact (alphabetical listing)

Measure	Notes
Allocated (max qty)	The number of installations (or users, depending on license type) that have specifically received allocations from the license in question. Allocations are essentially a manual linking of the installation/user with the license, to the exclusion of all other licenses, and may be made individually on the **Consumption** tab of the license properties. While they remain a 1:1 association, they can also be processed in bulk on the **Apply Allocations and Exemptions** page. *Related management view:* See either the **Consumption** tab of the license properties, or the **License Compliance > Apply Allocations and Exemptions** page.
Consumed (max qty)	The consumption count for each license, as filtered by your chosen dimensions and identified in the last exported compliance calculation for each point in time. This is the final consumption calculation for each license, taking into account all beneficial aspects like items covered by other product use rights, device exemptions and the like. *Related management view:* The license properties **Compliance** tab displays the **Consumed** count. Also visible in many license listings, such as **License Compliance > All Licenses**.
Covered by downgrade right (max qty)	The number of license entitlements consumed under a downgrade right (that is, the consumption occurred against a license for a later version or higher edition, but which included a downgrade right). *Related management view:* This collective total is not available in management views.

Measure	Notes
Covered by rights on VMs and hosts (max qty)	The number of virtual machines that are linked to the selected license, but that are not consuming any entitlements because they are covered by special product use rights. *Related management view:* For an individual license, look at the **Consumption** tab of license properties. Filter for **Inventory device type** of `Virtual machine`, and check the **Consumed** value (shows zero for any virtual device linked to this license but not consuming at the last license compliance calculation), and check for an **Exemption reason** such as `Covered by virtual application access`.
Covered by second use right (max qty)	The number of license entitlements that were *not* consumed because the related consumption was covered by the second use right on the license. *Related management view:* This collective total is not available in management views. For an individual license, open the license properties, and review the **Consumption** tab. A device contributing to this count shows `0` in the **Consumed** column, and an **Exemption reason** of `Second use`.
Exempted (max qty)	The number of installations linked to the license that do not consume a license entitlement because they have an exemption. Exemptions may take either of two forms: • An exemption for an individual inventory device linked to the license can be made on the **Consumption** tab of the license properties (these manual exemptions may also be made in bulk using the **Apply Allocations and Exemptions** page) • An exemption may be made automatically during license consumption calculations when the inventory device has been assigned a role (such as `Test`) that matches an exemption reason declared in the **Use rights & rules** tab of the license properties. *Related management view:* See either the **Consumption** tab of the license properties, or the **License Compliance > Apply Allocations and Exemptions** page.
Installed (max qty)	The number of installed software records that are linked to the license in question. This is unlikely to be the total number of installations of the same application. *Related management view:* For an individual license, look at the **Applications** tab of the license properties, and add the **Installed** column from the column chooser. There is no management view that shows this value for many licenses at a time. (Keep in mind that this is *not* the **Installed** count on an applications listing such as **Installed Applications**, because that figure is not filtered by license.)

Measure	Notes
Last purchase date	The most recent purchase date of all the purchases linked to the selected license. *Related management view:* For an individual license, you can check the result on the **Purchases** tab of the license properties, adding the **Purchase date** to the table from the column chooser, and sorting on the dates to see the latest one.
Licensed cores (max)	The number of processor cores that are covered by a license, totaled for all inventory devices linked to this license at the appropriate license consumption calculation. *Related management view:* In the properties for an individual license, check the **Consumption** tab and add the **Cores** column from the column chooser. This shows the number of cores licensed for each device attached to this license. (Remember that the core count for individual devices may be missing from inventory, depending on the inventory tool used; and that you can manually correct the value in the **Hardware** tab of the inventory device properties.) No management view of multiple licenses shows the total cores licensed.
Linked VMs (max qty)	The number of devices linked to this license that are virtual machines. This includes virtual machines which are not consuming entitlements because they are covered by other product use rights, exemptions, and the like. *Related management view:* This collective total is not available in management views.
Purchased (max qty)	The total number of license entitlements owned by the enterprise (or enterprise group) for the given license, being the sum of the **Licensed from PO** and **Extra entitlements** values stored in the properties of the license. This is the number of entitlements you are entitled to consume for this license. (Where individual purchases have been associated with particular enterprise groups, these facts are reflected in the subtotals for relevant groups.) *Related management view:* See the **Compliance** tab of the individual license properties, or review a license listing such as the **License Compliance > All Licenses** page.
Shortfall/Availability (max)	The calculated difference between **Total licensed** and **Consumed** (visible in the **Compliance** tab of license properties). Positive values show entitlements still available, and negative values show that consumption exceeds purchases (a purchasing shortfall). *Related management view:* For each license, see the **Shortfall/Availability** figure in the **Compliance** tab of the license properties. Also available in license listings such as **License Compliance > All Licenses**.

Measure	Notes
Total purchase price (max)	The total of all recorded costs for all the purchases linked to the particular license. (This may be unrelated to the costs of licenses consumed.) *Related management view:* The total price for each individual purchase is visible in the **All Purchases** page (and in the **Financial** tab of the properties of each purchase, where details can be recorded). There is no management view that shows the rolled-up total of all the purchases linked to a specific license.
Used (max qty)	Out of the installation records linked to the particular license, this is the subset for which there are usage records that meet the current criteria for software usage (by default, an application must have been used at least once in the last 3 months, with the settings adjustable on the **Usage** tab of the application properties). *Related management view:* You can inspect which devices/users are known to have used the licensed software on the **Consumption** tab of the license properties. As well, license views such as **License Compliance > All Licenses** offer the **Used** count for the installations linked to each license.

Software License Dimension

As part of the **Consumption analysis**, the **Software license** dimension allows you to drill down to an individual license for a specific product, as part of analyzing license consumption throughout your enterprise.

The **Software license** dimension expands into a four-level hierarchy that gives you access to (in order):

- **Publisher**
- **License Type**
- **Product Name**
- **License Name**.

Table 13: Attributes for the Software license dimension, Publisher level

Attribute	Notes
Publisher	The name of the company that publishes this application (and product). *Related management view:* For each application, the publisher is linked through the **General** tab of the application properties. However, the publisher must exist first for linking, and these records are maintained in the **Procurement > All Vendors** view. The publisher cited in each license is widely available in license listings, such as **License Compliance > All Licenses**.

Table 14: Attributes for the Software license dimension, License type level

Attribute	Notes
License type	The type of each license (as defined in FlexNet Manager Suite). Used here as a grouping mechanism, so that you must first choose the license type to drill down any further. *Related management view:* For each license, the **License type** appears in the **Identification** tab of the license properties. **License type** is also widely available in license listings, such as **License Compliance > All Licenses**.

Table 15: Attributes for the Software license dimension, Product name level

Attribute	Notes
Product	The common name for a family of applications, independent of version or edition details. This must be unique for any given publisher. *Related management view:* For each license, the **Product** name is displayed in the **Applications** tab of license properties (and you can link other applications there, each of which displays its own **Product** value). Keep in mind, too, that a license may be linked to applications from more than one product. The **Product (primary)** name is widely available in license listings, such as **License Compliance > All Licenses** (although sometimes you must add it to the listing from the column chooser).

Table 16: Attributes for the Software license dimension, License name level (alphabetical listing)

Attributes	Notes
Compliance status	Indicates whether or not use of software under this license complies with the license terms and conditions (and some related values). *Related management view:* For individual licenses, the **Compliance status** value is displayed in the **Compliance** tab of the license properties. The **Compliance status** column is widely available in license listings, such as **License Compliance > All Licenses**.

Attributes	Notes
Duration	The localized form of the possible license duration values, for which the English defaults are: • `Perpetual` (a license that is not time-limited) • `Subscription` (a license that must be renewed by regular payments, usually annually) • `Time limited` (a license that will expire, and typically cannot be renewed, such as a time-limited evaluation license). *Related management view:* For individual licenses, the **Duration** value is displayed in the **Identification** tab of the license properties. The **Duration** column is widely available in license listings, such as **License Compliance > All Licenses** (although sometimes you must add it to the listing from the column chooser).
Estimated unit price	An approximate cost for a single entitlement purchased for this license. If the **Override unit price** has been set on the **Purchases** tab of the license properties, this is the value used. Otherwise, the unit price (**Total price** divided by **Effective quantity**) is taken from the most recent purchase linked to the license. *Related management view:* For an individual purchase, the unit price is listed in the **Financial** tab of the purchase properties, as **Quantity X at unit price:**. The **Unit price (*currency*)** column is widely available in listings of purchases (although you sometimes you must add it to the listing from the column chooser).
Expiry date	For subscription (or other time-limited) licenses, this is the date when the current license expires. Any value is ignored when `SoftwareLicenseDurationID = 3` (that is, for a perpetual license.) A null value in this field also means that the license is perpetual, since it does not have an expiry date. *Related management view:* For individual licenses, the **Expiry date** value is displayed in the **Identification** tab of the license properties. The **Expiry date** column is widely available in license listings, such as **License Compliance > All Licenses** (although sometimes you must add it to the listing from the column chooser).

Attributes	Notes
Grants downgrade right	A Boolean that indicates whether this license offers downgrade rights. ***Tip:*** *There is no corresponding reporting on the upgrade right. This is because when the upgrade right is available, FlexNet Manager Suite automatically updates the primary application on the license to the most recent version. This means, in effect, that every license with the upgrade right has already been upgraded to the max.* *Related management view:* For license types that support downgrade rights, there is a **Downgrade rights** section in the **Use rights & rules** tab of the license properties that gives details. There is no equivalent of this Boolean in management views within FlexNet Manager Suite.
Grants rights on VMs and hosts	A Boolean that indicates whether the license provides specific rights for virtual machines or their hosts. *Related management view:* For individual licenses, when the **Use rights & rules** tab includes a section on **Right of second use**, the setting **No special virtualization rights** sets this Boolean to `0` (false). The remaining three settings on that section set this Boolean to `1` (true).
Grants second use right	A Boolean that indicates whether this license offers second use rights. Only supported for license types where **License type supports second use right** is `1` (true). *Related management view:* For license types that support second use rights, there is a **Right of second use** section in the **Use rights & rules** tab of the license properties that gives details. There is no equivalent of this Boolean in management views within FlexNet Manager Suite.
License	The full name of this license. *Related management view:* For each license, the **Name** is displayed (end editable) in the **Identification** tab of license properties. The **Name** is displayed in every license listing, such as **License Compliance > All Licenses**.
License edition	The edition of this license. While this is strictly an attribute of the license, a common convention is to allow automatic updating to reflect the edition of the latest application version linked to the license. *Related management view:* For each license, the **Edition** is displayed (end editable) in the **Identification** tab of license properties. The **Edition** is widely available in license listings, such as **License Compliance > All Licenses** (although sometimes you must add it to the listing from the column chooser).

Attributes	Notes
License status	Where this license was (at snapshot time) in the license life-cycle: whether `Purchased`, `Received`, `In stock`, `Active`, or `Retired`. ***Tip:*** *Do not confuse this value with* ***Compliance status****.* *Related management view:* For each license, there is a **Status** drop-down list available in the **Identification** tab of the license properties. The **Status** column is widely available in license listings, such as **License Compliance > All Licenses** (although sometimes you must add it to the listing from the column chooser).
License type supports second use right	A Boolean that indicates whether the current license type is capable of supporting the right of second use. *Related management view:* For individual licenses, the **Use rights & rules** tab includes a section on **Right of second use** when this value is true, and otherwise omits the section entirely. There is no equivalent available in management views.
License version	The version of this license. Note that this version number applies strictly to the license, and so may have been used in custom ways in your enterprise; but a common convention is to make the license version reflect the version of the application initially licensed here (although these versions may change through upgrade and downgrade rights). Another convention is to allow automatic updating to reflect the latest application version linked to the license. *Related management view:* For each license, the **Version** is displayed (end editable) in the **Identification** tab of license properties. The **Version** is widely available in license listings, such as **License Compliance > All Licenses** (although sometimes you must add it to the listing from the column chooser).
Subject to true-up	A Boolean that indicates whether this is a true up license. A true up license cannot go into breach, with any over-consumption in a period being adjusted through the regular (usually annual) true up process. *Related management view:* For each license, there is a **Subject to true up** check box available in the **Identification** tab of the license properties. The **Subject to true up** column is widely available in license listings, such as **License Compliance > All Licenses** (although sometimes you must add it to the listing from the column chooser).

Common Dimensions

This section documents dimensions that are common to both the **Installation** analysis and the **Consumption** analysis. These include:

- The **Point in time** dimension, which sets the reporting period for your custom reports, and determines how many saved data snapshots are included in them
- Four dimensions covering the four types of enterprise group (**Corporate unit**, **Location**, **Cost center** and **Category**) that allow you to segment the reported figures across your enterprise, using the corporate structure you have established in FlexNet Manager Suite
- Four near-duplicate dimensions (**Corporate unit (Software license)**, **Location (Software license)**, **Cost center (Software license)** and **Category (Software license)**), which, although they are unique to the **Consumption** analysis, are included in this section because their attributes and values exactly duplicate the previous four. The distinct purpose of these additional dimensions is to allow reporting on the *ownership* of software licenses, as distinct from the *consumption* of software licenses using the previously-mentioned four dimensions. Thus you could, for example, use custom reports to track down licenses assigned to (owned by) the Marketing department, but accidentally being consumed by the Sales team.

Point in Time Dimension

The **Point in time** dimension expands into a tree hierarchy with three levels:

- **Year**
- **Month** within a year
- **Day**, which is really the full date/time value when the snapshot was recorded in the data warehouse database.

You can, of course, pick any period (such as a year) to prepare your report, and then within the published report drill through to smaller time intervals, down to the individual data snapshot. Snapshots are recorded weekly.

This dimension has no equivalent in the management views within FlexNet Manager Suite, and does not reflect data stored in the compliance database.

Table 17: Attributes for the Point in time dimension

Attributes	Notes
Year	The year in which a snapshot was captured.
Month	The month in which a snapshot was captured.
Month-year	A convenience caption for reporting that combines the numeric month and year values. For example, August 2016 is represented as `8 - 2016`.
Day	The day (number within the month) on which the snapshot was captured.
Date	The full date/time value when the snapshot was stored (for example, `2016-09-03 14:40:00.000`). The date is represented in the extended format defined in ISO 8601:2004 (that is, `YYY-MM-DD`) and the time is in 24-hour format listing hours, minutes, seconds, and milliseconds. The time zone is not included, but all timestamps are recorded using the time setting on your central application server.

Enterprise Group Dimensions

There are four kinds of enterprise group in FlexNet Manager Suite:

- **Corporate unit**
- **Location**
- **Cost center**
- **Category**.

Each of these four kinds is available as a dimension for your custom reports, present in common in both the **Installation analysis** and the **Consumption analysis**. This allows you to segment either application installation numbers or license consumption numbers across various kinds of enterprise groups, as best fits your corporate approach to group management.

Tip: *Keep in mind that the numbers quoted for each enterprise group are "rolled up totals": that is, they are the total for all the children of this group, plus any local value for the group itself. For example, consider this simplified location hierarchy (each row being the only child of the one above) with the local installation counts of a particular application as shown. Each location then shows the rolled-up total installation values given in the third column:*

cation	*Local installations*	*Rolled-up total shown*
North America HQ	*12*	*55*
North-west Region Office	*10*	*43*
Chicago Office	*33*	*33*

In reality, the rolling up of totals is more complex, since any high-level enterprise group is likely to have many peer children, each of which has many peer children, and so on down through the tree.

In the case of the **Consumption analysis** (only), as well as the basic enterprise groups that you may use to analyze consumption across different groups, there is a second set of the same groups with a slightly different naming convention:

- **Corporate unit (from license)**
- **Location (from license)**
- **Cost center (from license)**
- **Category (from license)**.

These identify the same kinds of groups, but in this case as an attribute of the license itself, identifying any relationship between the license and enterprise groups established on the **Ownership** tab of the license properties.

The presence of enterprises groups in these two ways, both tracking the *ownership* of the license and separately segmenting the *consumption* of the same license, allow you to probe scenarios such as licenses that you thought were assigned to one group being consumed elsewhere (which means the license was *over*-assigned to the first group, and has spare capacity).

Each of the enterprise group dimensions expands to show a tree hierarchy with maximum depth of 10 levels of child groups of the same kind. Each level is identified in the group's label, such as **Location - level 3**.

At each level, each corporate group has only a single attribute available:

Table 18: Attribute for the enterprise group dimensions, at each of the 10 levels

Attribute	Notes
Enterprise group **name**	The name of each Corporate unit, Location, Cost center, or Category, as displayed in the label (in place of the *Enterprise group* placeholder). *Related management view:* The name of each enterprise group is available wherever groups are included in management views within FlexNet Manager Suite. The details are maintained in the listing for each enterprise group (available through **Enterprise** > ***Group type***), where you may expand the hierarchy to any focus point, and either click the **+** icon to add a new child, or click the edit (pencil) icon to modify the name of an existing group.