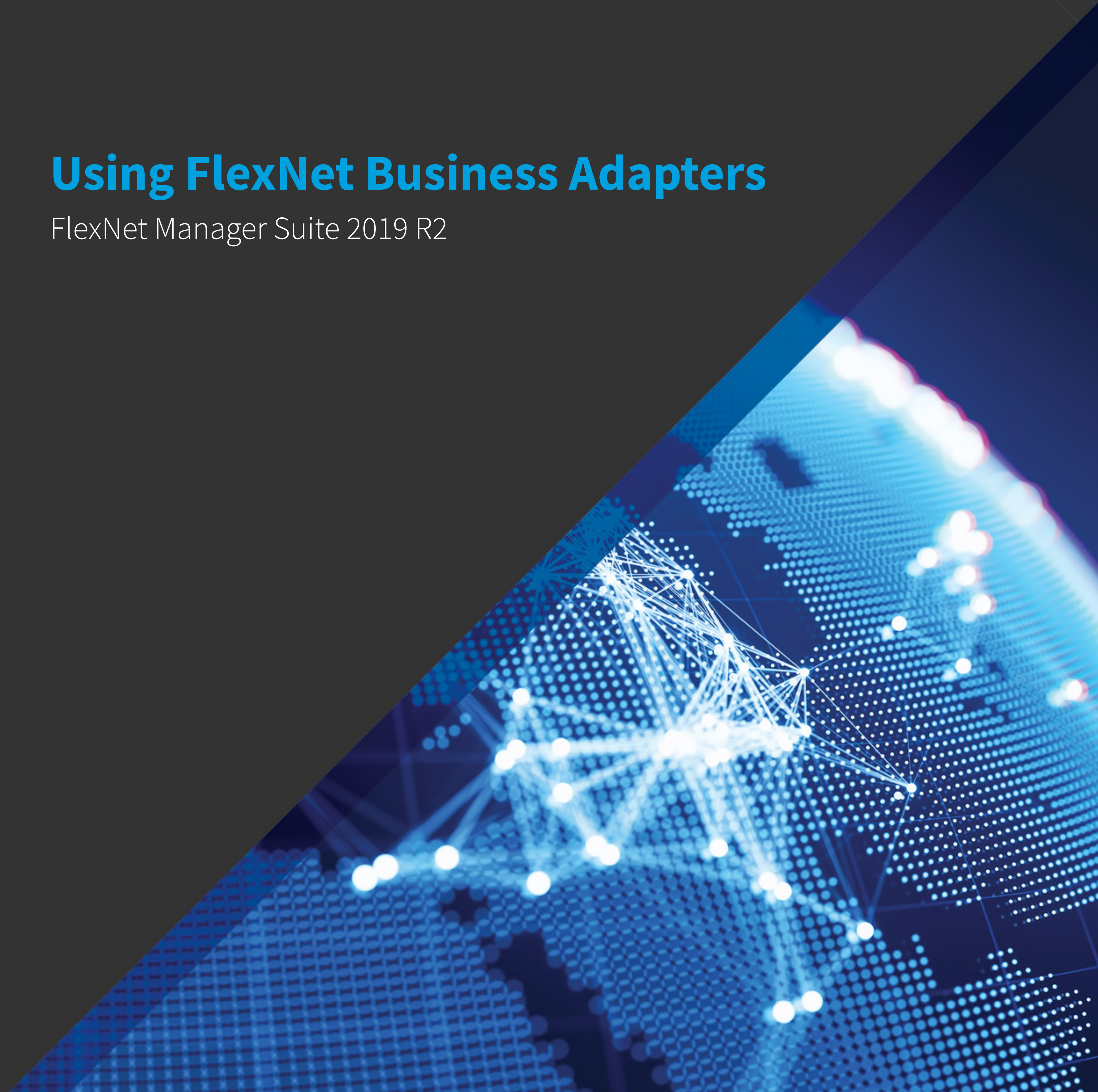
Using FlexNet Business Adapters

FlexNet Manager Suite 2019 R2



Legal Information

Document Name: Using FlexNet Business Adapters version 2019 R2 (for on-premises implementations)

Part Number: FMS-14.0.1-BI01

Product Release Date: December 16, 2019

Copyright Notice

Intellectual Property

Restricted Rights Legend

Preface

"Business adapters" are ways of connecting to data sources in your enterprise and extracting relevant data for import into FlexNet Manager Suite. The name means that:

- They deal with *business* data, and specifically *not* inventory of software and hardware (those kinds of data are handled by *inventory adapters*)
- They *adapt* the business information from the formats where it is normally stored into the formats needed by FlexNet Manager Suite.

This document covers two separate aspects of using business adapters:

- Part I covers use of the FlexNet Business Importer, a utility for importing data from common business sources into FlexNet Manager Suite. These imports make use of existing business adapters, whether these are ones that you have developed previously, or ones built for you by a consultant. Examples of common business data include:
  - Purchase records
  - HR data, including organizational structures
  - License data
  - Contract information
  - Publishers and suppliers.
- Part II, for more advanced readers, covers creation of new business adapters using a tool called the Business Adapter Studio. This tool specifies both the connections to external data sources, and the mapping of external to internal field names. (The FlexNet Business Importer may also be triggered automatically by the Business Adapter Studio for testing adapters as you develop your adapters.)

For brevity throughout this document, the FlexNet Business Importer is simply called the Business Importer. Unless otherwise clarified, all discussion of 'adapters' in this document relate to business adapters (as distinct from inventory adapters).

***Important:** Beware of using a business adapter to set properties that are also set by built-in import functionality. If you set these properties through a business adapter, and you use a different value than the one imported by default, the property value toggles back and forth between the two values, depending on which import type ran most recently. Some examples of properties that can be updated in conflicting ways are:*

- *Several properties of assets, when the asset has been linked with an inventory device from which some properties are imported.*

  ***Tip:** For more about links between assets and inventory devices, see Customizing the Asset-Device Linkage.*

  Commonly affected are these properties from the `Asset` table in the compliance database:

  - *`Asset.ShortDescription`*

    - *`Asset.ModelNo`*
    - *`Asset.Manufacturer`.*
- *`ComplianceUser.UserName`, which is imported from Active Directory on Windows devices.*

Contents

I

The FlexNet Business Importer

The FlexNet Business Importer is a command-line tool, also executed by the Business Adapter Studio, that imports data from a variety of sources through customizable business adapters into the central operations databases (specifically, the compliance database) of FlexNet Manager Suite.

The Business Importer focuses on the import of business-related data, such as purchases, contracts, organizational structures, asset registers, and the like.

Excluded from 'business data' is the software and hardware inventory gathered from the computers within your enterprise. This exclusion means, for example, that inventory `Evidence` and linked `Application` objects that result from inventory imports are not accessible through the FlexNet Business Importer.

Tip: *If you are seeking to import inventory details, there are other tools specific to that purposes:*

- *Built-in inventory adapters that form a standard part of the FlexNet Manager Suite*
- *Custom inventory adapters built through the Inventory Adapter Studio*
- *Import of inventory spreadsheets, either as a one-time import through the web interface of FlexNet Manager Suite, or as scheduled imports through an inventory beacon (see the Importing Inventory Spreadsheets and CSV Files chapter in the FlexNet Manager Suite System Reference PDF, available through the title page of online help).*

This part covers direct use of the FlexNet Business Importer as a command-line tool.

Note: *The functionality of the FlexNet Business Importer is increased when it is executed on your central application server, where it has direct access to the compliance database. This possibility is available only for on-premises implementations of FlexNet Manager Suite, and is not available for cloud-based implementations. In both kinds of implementation, it is also operable in disconnected mode, running on an inventory beacon, where the import is saved to an intermediate file, automatically uploaded to the central application server, and imported into the compliance database.*

The chapters in this part cover:

- Prerequisites for using the Business Importer, and its command-line options for running manually or through a scheduled task.
- The format or structure of the XML files that configure each business adapter.
- The content of a series of default CSV templates that, together with matching XML business adapter files, provide a

starting point for editing your own business adapters.

- Tips and guidelines for working with different kinds of data sources.
- The data model that is available to the Business Importer (and therefore the Business Adapter Studio), together with its matching entities in the central operations databases for FlexNet Manager Suite. This data model permits access to only a subset of the complete database (for all details about the structure of the central database itself, see the companion volume FlexNet Manager Suite Schema Reference).

This part of the document is for technically competent people who are possibly hand-editing business adapters (or using business adapters that have already been completed and tested) and are executing the Business Importer directly from the command line. Such expert readers are comfortable with:

- Structured Query Language (SQL)
- Running queries against tables and columns
- Reading schema documentation
- XML mark-up.

***Warning:** All imports modify the FlexNet Manager Suite database. Incorrect settings for the Business Importer or in business adapters may result in deleting, modifying or overwriting important data. It is mandatory to back up target databases before executing the Business Importer with changed settings or a changed business adapter. This backup enables a rollback of the database to its original state, if the incoming data is corrupted or the Business Importer misconfigured. It is also highly recommended to test any modifications to an existing business adapter, or test a new business adapter, against a preproduction environment before moving it into production.*

1

Prerequisites

Availability (Without Installation)

The Business Importer is available as a standard part of the product installation:

- It is installed on each inventory beacon, along with the Business Adapter Studio. The default location on each inventory beacon for the Business Importer executable is `C:\Program Files\Flexera Software\Inventory Beacon\DotNet\bin\MGSBI.exe`.
- For on-premises implementations only, the Business Importer is also available on the central application server for use where direct connection to the central compliance database is available. There are two possible locations:
  - If you have an inventory beacon co-installed on your application server, the Business Importer is available in the same path as shown above for all inventory beacons
  - If you do not have a co-installed inventory beacon, the default location is on the server hosting the batch server functionality, under `C:\Program Files (x86)\Flexera Software\FlexNet Manager Platform\DotNet\bin\MGSBI.exe`.

Because Business Importer is installed as standard, it does not require additional specific installation tasks.

In all cases, the complete Business Importer consists of:

- The executable, called `MGSBI.exe`
- A set of template XML files for you to modify as required
- This documentation is available through the title page of online help.

Connections and Drivers

Connections can be considered in two directions:

- 'Downstream' from the Business Importer to the original data sources
- 'Upstream' from the Business Importer as it uploads the collected data to the central application server for FlexNet Manager Suite.

Downstream, it's obvious that the Business Importer requires connections to the source data it is to collect. Depending on the data source used, additional drivers may be required. For instance, if data is read from Oracle databases, the Oracle OLE-DB driver (delivered by Oracle as part of the Oracle client) must be present .

In contrast there are no special requirements needed 'upstream':

- In disconnected mode, where the Business Importer is running on an inventory beacon and cannot access the central compliance database, no drivers are required. When it is executed by the Business Adapter Studio, the Business Importer writes the collected data to a special staging file on the inventory beacon, where other automated processes upload it to the central application server, and subsequently import it into the compliance database.
- In connected mode, which (clearly) requires a SQL Server connection from the Business Importer instance to the database server hosting the central compliance database, there is no need to install any specific SQL Server tools or drivers. The Business Importer uses the Microsoft SQL OLE-DB driver provided as standard on appropriate Microsoft Windows Server operating systems.

2

Command Line for Business Importer

These options are available when the Business Importer is run from the command line.

Syntax:

Syntax

InstallationPath\ **`MGSBI.exe`** / *SwitchName*[= *Value*] [/ *SwitchName*[= *Value*] ...]

In normal use, exactly one of the following switches must be specified:

- `/Import`
- `/Query`
- `/Simulate`.

However, the `/Test` switch may be used alone, without requiring any of the above. The full set of switches is shown below.

Examples

In this example, the Business Importer is run in connected mode on the central application server of an on-premises installation. The adapter file is in the current directory. Logging details are specified in the adapter file. (Text is wrapped for documentation, and should be entered on a single line.)

```
"C:\Program Files\Flexera Software\Inventory Beacon\
DotNet\bin\MGSBI.exe"
     /ConfigFile="My Adapter.xml" /Import=Purchases
```

This example shows the Business Importer running on an inventory beacon in disconnected mode. The adapter file under test is in a subdirectory below the executable, and the output is redirected to an intermediate file for review.

```
cd C:\Program Files\Flexera Software\Inventory Beacon\
DotNet\bin\MGSBI.exe
     /ConfigFile=adapters\productionImports.xml
     /Import=EnterpGroups /Write="C:\temp\DataReview.xml"
```

***Note:** This output file is not in a format suitable for upload directly to the central server in a cloud implementation. The upload format requires:*

- *An upload manifest*
- *A copy of the adapter XML file*
- *This data file.*

All must be zipped into a single archive for automatic upload, and this is not possible using the Business Importer alone. However, after reviewing the adapter's output with the command-line shown here, you could save the adapter XML file, and schedule its execution through the inventory beacon interface. The inventory beacon automatically provides the additional data and archiving for upload.

Options

The supported set of parameters and switches (in alphabetical order) is as follows.

***Tip:** Parameter/switch names are case insensitive.*

Parameter/Switch	Switch Values	Details
`/?` or `/Help`		Displays a list of available parameters and related material.
`/ConfigFile=`	*`AdapterXMLFile`*	Specifies the path and file name of the adapter XML file that defines connection details and column mapping for the particular import. If the file path or file name contain any spaces, enclose the entire value in double quotation marks. If this switch is omitted, the Business Importer searches for a file called `MGSBI.XML`, in the same folder from which the Business Importer is executing. If a file name is defined without a path, the Business Importer looks in its execution folder; and relative paths are also relative to the execution folder. Where neither the switch nor the default file is present, or when the specified file cannot be found, execution terminates with an error.

Parameter/Switch	Switch Values	Details
`/Encrypt`		Encrypt all database connection strings found in the *`AdapterXMLFile`*, rewriting the XML file with the new values replacing the originals. This applies not only to the central compliance connection, but also to all other data source (Import) connections. This protects the connection information in case the adapter XML file is accessed by a hostile entity. Connection strings, once encrypted, cannot be decrypted for display in the XML file as plain text. If you need to update an encrypted connection string in the XML file, remove the encrypted content, and restore a plain text version of the connection string. Running the Business Importer again with this switch will re-encrypt the plain text connection strings in the XML file. Encryption uses AES cryptography, and the encryption key is specific to the computer on which the encryption is run. ***Important:*** *Adapter XML files containing encrypted connection strings cannot be used on any other computer except the one on which encryption is performed. Adapters must be encrypted separately on each computer on which they are used by the Business Importer.*
`/Import=` or `/Import`	*`ImportName`*	Collect data from the data source specified in *`ImportName`* within the *`AdapterXMLFile`*, and load it into the database for which (in connected mode) the connection string is also specified in the *`AdapterXMLFile`*. The import name is case-sensitive. The second form (`/Import`), without the equals sign or a name of an import, may only be used when there is a single import in the *`AdapterXMLFile`*. If this empty switch is used when the *`AdapterXMLFile`* specifies multiple imports, the import fails with an error. ***Tip:*** *Only a single import may be run from each command-line invocation.*

Parameter/Switch	Switch Values	Details
`/Log=`	One of: `Silent`, `Critical`, `Errors`, `Warnings`, `Information`, `Debug`, `Default`	Select the kind of logging that should be output to the console. • With **Silent**, no log file is written for this pass (even if a log file has been specified). • **Critical**, **Errors**, **Warnings**, **Information**, and **Debug** filter the logging so that nothing less severe is displayed. Listed here from highest to lowest priority, any setting lists entries of that level and *higher* priority. For example, if you set `/Log="Errors`, the log includes errors and critical messages. • **Default** (which is also the behavior when this switch is not specified) sets the level to **Information**, meaning that it displays **Information** and higher messages. Note that either the `/LogFile=` switch for the Business Importer must be specified, or the *`ImportName`* defined in the *`AdapterXMLFile`* must specify a log file, for a log file to be written.
`/LogFile=`	*`LogFile`*	Path and file name where an additional log file may be written. Alternatively, the `<log/>` element may be specified in the *`AdapterXMLFile`*. If both are omitted, no log file is written (there is no default). If different values are given in the command line and the *`AdapterXMLFile`*, both files are written. ***Tip:*** *This log file does not respect the setting in the `/Log=` command-line switch, and always logs `Information` and higher messages.*
`/OperatorLogin=`	*`AccountName`*	Used by FlexNet Manager Suite to create history records showing which operator made changes to which database elements.

Parameter/Switch	Switch Values	Details
`/Query=`	*ImportName*	Run the query specified in the import definition (in the *AdapterXMLFile*) against the data source and list the individual records on the console in CSV format. The case-sensitive import name is mandatory when this switch is used.
`/Read=`	*DataFilePathAndName*	Over-rides the connection details specified in the *ImportName* within the *AdapterXMLFile*, and instead the adapter reads data from the intermediate data file specified. This intermediate file format reflects the data structure defined in the import element *ImportName* in the *AdapterXMLFile*. This switch is for internal use in disconnected mode, where the Business Importer has written the data to the intermediate file on an inventory beacon, the intermediate file has been automatically uploaded to the central server, and now the Business Importer resumes the process by reading from the intermediate data file and continuing the import process.
`/Simulate=`	*ImportName*	Runs a full import (similar to the `/Import` switch), within a single database transaction. Afterwards, the transaction is rolled back so that nothing from the import persists. It outputs to the console a list of any data information messages, warnings, and errors, together with a summary count (per object type) of records matched, create, updated, deleted, and rejected. The case-sensitive import name is mandatory when this switch is used.
`/Test`	*ImportName*	Test the downstream connection to the data source. The case-sensitive *ImportName* is mandatory when this switch is used, and identifies which connection within the *AdapterXMLFile* is tested. Use the special case of `/Test=ManageSoft` to test the upstream connection to the FlexNet Manager Suite compliance database. ***Important:*** *This special case is available only for on-premises implementations.*

Parameter/Switch	Switch Values	Details
`/Write=`	*`DataFilePathAndName`*	Over-rides any upstream connection details specified in the *`AdapterXMLFile`* for the central compliance database, and instead the adapter writes collected data to the intermediate data file specified. This intermediate file format reflects the data structure defined in the import element *`ImportName`* in the *`AdapterXMLFile`*. This switch can be used either for testing, to inspect the data written; or on-premises implementations, it can be used in disconnected mode. The output file may be manually transferred to your central application server, and used with a command-line import there.

Obsolete switches

The following switches are now obsolete, and must not be used:

- `Turbo=[true|false]`
- `NewMode=[true|false]`
- `Boost=[true|false]`.

These switches were previously used to put legacy versions of the Business Importer into turbo mode. This is the only mode in which the Business Importer now operates, making the switches redundant, and they have been obsoleted. Ensure that, if you are migrating legacy adapters to a current version of the Business Importer, any of these switches are removed from command-line scripts and the like. Attempting to set any of these switches to `false` now produces an error similar to the following:

```
ERROR: The legacy import mode (switchName=false) is no longer supported
```

3

Adapter XML Files

The Business Importer is driven by XML files that adapt the data formats in the source to suit the target database (normally, the central FlexNet Manager Suite compliance database). Each of these XML files defines an 'adapter' that includes one or more 'imports', and specifies:

- Connection strings to source data, and (in connected mode) to the target database as well
- Queries required to fetch source data
- Data mapping between source and target.

Sample adapter files are installed by default on each inventory beacon at `C:\ProgramData\Flexera Software\Beacon\DDI`.

The format of the XML files is straight-forward, accepting only the following seven XML elements:

XML Element	Purpose
`root`	Required. Container element for all other elements.
`ManageSoft`	Required. Specifies details about the target account and (in connected mode) target database connections. Each adapter XML file must contain exactly one `<ManageSoft>` element.
`Imports`	Required. Container element for a list of `<Import>` elements.
`Import`	A functional adapter XML file must contain one or more `<Import>` elements. Each defines the connection to a data source, the database objects to be imported, and the logging required. Only one `Import` may be run at each command-line invocation.
`Log`	Optional. Defines the log files associated with each import. (A log file may also be defined in the command line for the Business Importer, described in Command Line for Business Importer.) This element is available by default for a business adapter running in connected mode on your central application server, and can be manually added for adapters running in disconnected mode on an inventory beacon.
`Object`	At least one `<Object>` element must be present as a child of each `<Import>` element. Contains one or more `Property` elements, and the processing rules associated with an object in the central compliance database.

XML Element	Purpose
`Property`	Defines the mapping of incoming data to a single property of a database `<Object>`.

The following sections cover these elements in detail.

root Element

This is the container for all the operational elements in the adapter XML file.

Example

```
<root>
    <ManageSoft ... />
    <Imports>
        ...
    </Imports>
</root>
```

Contains

Child	Comments
`ManageSoft`	Mandatory. Exactly one may exist in the adapter XML file. Must be the first child of `<root>`.
`Imports`	Mandatory. Exactly one may exist in the adapter XML file.

Contained by

Nothing. This is the root element defining the body of the XML document.

Supported attributes

None.

ManageSoft Element

This element contains all the information needed for connection to the central compliance database.

Examples

Running on the central application server in an on-premises implementation, using the database connection string specified in the registry, and executing separate updates without wrapping them in a single transaction for speed (as distinct from a single database transaction, where rollback is supported in the event of any failure to the updates):

```
<ManageSoft ConnectionType="ECM" RunInTransaction="False" />
```

In connected mode, when access is available to the central compliance database using Windows authentication, and the connection string has not been encrypted (the connection string attribute should be all on one line):

```
<ManageSoft AccountIsEncrypted="False"
    ConnectionString="Integrated Security=SSPI;Persist Security
Info=False;Initial Catalog=FNMP;
    Data Source=WIN2K8SRV2"/>
```

In disconnected mode, when the adapter is exercised on an inventory beacon with no access to the central compliance database:

```
<ManageSoft ConnectionType="Default" RunInTransaction="False" />
```

In connected mode, when access is available to the central compliance database but the connection string has been encrypted (which locks the adapter file to use on this same computer). The encrypted connection string has been shortened in this example for printing purposes.

```
  <ManageSoft ConnectionType="Specific"
      ConnectionString="1|2Y728ZndMMMpMwQMc...WikVzdsnJvxL"
      RunInTransaction="False" AccountIsEncrypted="True" />
```

Contains

`<ManageSoft />` is an empty element, containing no child elements. (See attributes listed below.)

Contained by

`<root>` is the parent.

Supported attributes

Attribute	Details
`AccountIsEncrypted`	Mandatory. Boolean: • `True` when the `ConnectionString` has been encrypted • `False` when the `ConnectionString` is interpreted as plain text. (Use this value also in disconnected mode.) For details about encrypting connection strings in an adapter XML file, see Command Line for Business Importer. ***Note:*** *The* `AccountIsEncrypted` *attribute is used by the Business Importer to determine whether the* `ConnectionString` *value is encrypted. Setting the value to false once the connection string has been encrypted results in a connection error when the Business Importer is run with this adapter file.* ***Tip:*** *To remove encryption, you must do all of the following three things:* • *Set* `AccountIsEncrypted` *to false.* • *Replace the encrypted string value for* `ConnectionString` *with a valid plain text version.* • *Repeat the plain text replacement for all other connection strings defined for each* `<Import>` *in the XML file. (The single use of the* `/Encrypt` *switch on the command line for the Business Importer also encrypts all* `ConnectionString` *values for each import object in the adapter XML file.)*

Attribute	Details

`ConnectionString`	Defines the OLE-DB connection string to the central compliance database. This value may be plain text (when `AccountIsEncrypted` is `False`), or may contain unreadable character data when the connection string has been encrypted (`AccountIsEncrypted` is `True`). For details about encrypting connection strings in an adapter XML file, see Command Line for Business Importer.
	The attribute is mandatory in connected mode (with access to the central compliance database) when `ConnectionType` is omitted or has the value `Specific`; and is forbidden in disconnected mode (where data must be written to an intermediate file on an inventory beacon for subsequent upload and import).
	The unencrypted form of the connection string value depends on which type of authentication is used to connect to the central SQL Server database:
	• For Windows authentication:
	`ConnectionString="Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=`*`databaseName`*`;Data Source=`*`SQLServerName`*`"`
	• For SQL authentication:
	`ConnectionString="Password=`*`SQLPassword`*`;Persist Security Info=True;User ID=`*`SQLAccount`*`;Initial Catalog=`*`databaseName`*`;Data Source=`*`SQLServerName`*`"`
	In either case, substitute your values for the following placeholders as required:
	• *`SQLServerName`* — The server name or IP address (or dot for connection on the current server) of the SQL Server hosting the central compliance database. Using the server name requires that a DNS is accessible and operating, to resolve the server name.
	• *`databaseName`* — The name of the central compliance database. (In a default installation, this database name is commonly `FNMSCompliance`; or in an upgraded implementation, it may be `ManageSoft`.)
	• *`SQLAccount`* — The user name (or account) permitted access to the central compliance database.
	• *`SQLPassword`* — The password for the above account, needed to complete the credentials for accessing the compliance database.
	Tip: *If you do not wish to type in a value for `ConnectionString`, you can build a connection string using a UDL file. For more information, see Validating Connection Strings.*

Attribute	Details
ConnectionType	Optional. Defines the method for determining the connection. Valid values are: • Default or Specific — One of these synonyms is the only permitted value when the Business Importer is running on an inventory beacon in disconnected mode (Default may be best practice here, being easier to understand in the context). In this case, no ConnectionString can be provided, since there is no connection to the target database. For an on-premises implementation, when the adapter is running in connected mode on an application server, either of these values requires that the ConnectionString value is used to connect to SQL Server to access the operations databases. • ECM — The ConnectionString value is ignored, and connection is made using the details stored in the registry at HKLM\SOFTWARE\ManageSoft Corp\ManageSoft\Reporter\CurrentVersion\DatabaseConnectionString. This registry is only populated (and the ECM value is only valid) when the Business Importer is running on the central application server for FlexNet Manager Suite, in an on-premises implementation.
RunInTransaction	Optional. Boolean: • True — All imports run from this adapter XML file are wrapped as a single transaction in the central database. This allows for transaction roll-back in the event of any difficulties. • False — Data transactions (insert, update or delete) on the central compliance database are actioned individually. This reduces the risk of issues around table locking during business imports disrupting other processes. When this attribute is not specified, the default depends on where the adapter is running: • In an on-premises implementation when the Business Importer is running on the central application server, the default is True. • For adapters created in disconnected mode on an inventory beacon, RunInTransaction is set to false.

Imports Element

This is a simple container element for the Import elements within the adapter XML file.

Example

```
<Imports>
    <Import ... />
```

```
</Imports>
```

Contains

Child	Comments
`Import`	Mandatory. One or more `Import` elements are required.

Contained by

`<root>`.

Supported attributes

None.

Import Element

This defines downstream connection details, business rules, and data mapping for a particular data import.

Example

```
<Import Name="Asset"
        Type="CSV"
        UsePhysicalTables="false"
        TraceField="AssetName"
        TraceAction="Rejected" TraceLifeTime="1 month"
        ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=.\Data;Extended
Properties='text;HDR=Yes;FMT=CSVDelimited'"
        AccountIsEncrypted="False"
        Query="select * from [asset.csv]">
      <Log ... />
      <Object ... >
        ...
      </Object>
</Import>
```

Contains

Child	Comments
`Log`	Optional. Used to direct logging output for this `Import` to a specific destination. Alternatively, logging may be directed by a command-line switch for the Business Importer (see Command Line for Business Importer). Where both the command-line switch and the `Log` element exist, and direct output to different locations, both logs are written.

Child	Comments
Object	Mandatory. Each <Import> must contain at least one database <Object> to which imported data is directed; and typically each will contain several <Object> elements. The ordering of the Object elements is important, as they are processed in the same order that they appear in the adapter XML file. Therefore (for example), if an adapter imports purchases that reference new vendors, it is important that the Object element for vendors appears earlier in its parent Import list than the Object element for the purchases. If this ordering is wrong, the overall processing may fail when a purchase (processed too early) fails to reference the vendor that has not yet been created.

Contained by

Imports.

Supported attributes

The large number of attributes available on the Import element are divided by their purpose, as follows:

- General purpose attributes
- ADSI specific attributes (Active Directory Service Interfaces are used to gather information from Active Directory)
- Text file handling
- Web service settings.

General purpose attributes

These attributes apply to all types of imported data.

Attribute	Details
AccountIsEncrypted	Mandatory. Boolean: • True when the ConnectionString to the downstream data source has been encrypted • False when the ConnectionString for this Import is interpreted as plain text. For details about encrypting connection strings in an adapter XML file, see Command Line for Business Importer. **Note:** *The AccountIsEncrypted attribute is used by the Business Importer to determine whether the ConnectionString value is encrypted. Setting the value to false once the connection string has been encrypted results in a connection error when the Business Importer is run with this adapter file.*

Attribute	Details
`BulkCopyBatchSize`	Optional. Sets the number of records transferred in a batch for the data loading to the temporary table (in connected mode) or intermediate data file on the inventory beacon (in disconnected mode). When the attribute is omitted, the default value is `1000`. Special value: • `0` — Load all data in a single batch.
`BulkCopyTimeOut`	Optional. Sets the number of seconds after which an incomplete data load to the temporary database table times out. When this attribute is omitted, the default value is `3600` (1 hour). This value may be increased for extra-large data sets.
`CleanUpControlCharOn`	Optional. A list of field names in the incoming data for which illegal characters (those with an ASCII value below 32) are removed and replaced with an underscore character. In typed data, the clean-up is applied only to text fields. If this attribute is omitted, no clean-up occurs. Notes for specifying the list: • Individual entries in this list must match column names returned by the query defined in `Query` attribute. • The list uses a semi-colon as a separator between column names. • The asterisk character (*) is supported as a wild card matching all text columns in the data set. • The caret character (^) is supported for specifying a single exception. Examples: • `CleanUpControlCharOn="*"` — Cleans up all text columns. • `CleanUpControlCharOn="*;^Name"` — Cleans up all text columns except the `Name` column
`ConnectionString`	Mandatory. Defines the connection string for the downstream data source. This value may be plain text (when `AccountIsEncrypted` is `False`), or may contain unreadable character data when the connection string has been encrypted (`AccountIsEncrypted` is `True`). For details about encrypting connection strings in an adapter XML file, see Command Line for Business Importer. This attribute is mandatory, but its format depends on the value of the `Type` attribute: each connection type has its own format for the connection string. For details, see ConnectionString Attribute.
`DataTableName`	Optional. Sets the name of the physical database table that first receives the external data during import. This value is used only when `UsePhysicalTable=true`. See the `UsePhysicalTable` attribute for more information.
`ManageSoftTimeOut`	Optional. An integer value to set the timeout used by the Business Importer for each SQL query run against the central destination database. Special value: • `0` — (Default value.) Do not set any timeout limit.

Attribute	Details
`Name`	Mandatory. Uniquely identifies the name of the import in the adapter XML file. Special characters and spaces should not be used.
`Query`	Mandatory. The query used against the downstream data source. The format of the query depends on the `Type` of this `Import`. If the `Type` is `XML`, the query must be a null string. For details, see Query Attribute.
`Signature`	Optional. Database objects in the central compliance database have history properties that track who created the record and who last updated it. The properties are set by this attribute. Two case-sensitive keywords are supported in this attribute's value: • `[IMPORT NAME]` — Causes the history properties to contain the value of the `Name` attribute for this `Import` element. • `[USER NAME]` — Causes the history to contain the Windows login name (in the format *`domainName`*`\`*`userName`*) of the account running the Business Importer. Plain text values may used as alternatives if you need special values. If this attribute is omitted, the default values used is `"[USER NAME] ([IMPORT NAME])"`.
`Timeout`	Optional. Integer value that sets the maximum time in seconds that the Business Importer waits for the external query to execute against the downstream data source. The following special values apply: • `-1` — (Default value.) Use the time out determined by the source data connection (such as a source database timeout). • `0` — No time out.
`TraceActions`	Optional. Set what action(s) will be recorded in the detailed log stored in the target database. Valid values are: • `Creation` • `Deletion` • `Update` • `Rejected`. You may use a comma-separated string to list multiple actions: `TraceActions="Creation,Deletion,Rejected"` If this parameter is omitted, no actions are recorded in the tracing (even though `TraceField` is declared). Both settings are required if you wish to generate detailed tracing of actions in `ECMImportLog_Detail`. (Independently, the trace of which import has run is recorded in `ECMImportLog_Summary`, and the objects updated, inserted, and deleted are logged in `ECMImportLog_Object`. For more information about this detailed logging, see Detailed Tracing.)

Attribute	Details
`TraceField`	Optional. Identifies the input data property that is used to fill the description field in the detailed actions log stored in the target database. For more information about this detailed logging, see Detailed Tracing. Valid values are: • A best identifying column name, enclosed in brackets [ ], from the incoming data set • Combination of informative text and one or more column names, with each column name enclosed in brackets. For example: `TraceField="Record [ColumnA] – [ColumnB] has been processed"` ***Note:*** *Anything enclosed in brackets is treated as a column name while all else is simply text and appears as it is in the final* `RecordDescription` *field. In fact, if any SQL operators or functions are used above, they will appear as text in the resulting column and will not execute.* ***Tip:*** *For an adapter running in connected mode on the application server, this field is set to the appropriate natural key for the chosen object. For business adapters in disconnected mode on the inventory beacon, the field is left blank for you to populate.*
`TraceLifeTime`	Deprecated from release 2015 R2. The lifetime of tracing records is now set through the web interface for FlexNet Manager Suite (navigate to the system menu (⚙▼ in the top right corner) > **System Settings**, select the **Inventory** tab, and scroll down to the **Activity logs** section). The default value is 30 days. ***Note:*** *Trace records can multiply very quickly, blowing out the size of the target database. It is strongly recommended that where tracing is used, you specify the shortest convenient time for the tracing records to persist, consistent with your required business processes.*

Attribute	Details
Type	Mandatory. Specifies the kind of data source from which data is being imported. Valid values are: • SqlServer • Oracle • Excel • CSV • OleDB • ADSI • ODBC • WebService • XML.
UsePhysicalTable	Optional. Boolean: • true — External data is stored as a physical table in the database during import. Use this when you anticipate large data sets for this import. The physical table is created with the name specified in the DataTableName attribute if present, and is otherwise named ECMImport_*ImportName* (where *ImportName* is the value of the Name attribute of this Import element). ***Note:** The table is dropped and re-created each time the import is run.* • false (the default) — External data is stored as a temporary table in computer memory. The name of the temporary table is #ECMImport_*ImportName*. (The DataTableName attribute is ignored in this case.)

Attributes governing imports from Active Directory

Use these attributes to control collection of data using the Active Directory Service Interfaces. Note that where applicable, the same attributes may be used to control data gathering from other services, such as Novell eDirectory.

Attribute	Details
`PageSize`	Mandatory for eDirectory users (and otherwise ignored). An integer value to set the page size in a paged search. The default value is 0, which means do not do a paged search. If you are using Novell eDirectory, the `PageSize` attribute must be set to `-1`: `PageSize="-1"` Otherwise, paged searches are not yet supported by the Business Importer. Reserved for future development.
`ClientTimeout`	Optional. An integer value to set the number of seconds that the Business Importer waits for the server to return results. Special value: • `-1` — (Default value.) Wait indefinitely — no time out.
`Path`	Optional. When this attribute is not specified, the default is an empty string. Specifies the path to the desired object within the hierarchy of the directory service, and may vary according to the directory services in use. Example: `LDAP://CN=Users,DC=dsprovider,DC=nttest,DC=microsoft,DC=com`

`PropertiesToLoad`
(synonym `Properties`)
	Mandatory. A comma-separated list of properties to load from the directory service. Example:
	`PropertiesToLoad="distinguishedname,sn,cn,givenname"`
	Tip: *Either form of the attribute name may be used. This is equivalent to the above example:*
	`Properties="distinguishedname,sn,cn,givenname"`

`ReferralChasing`	Optional. Sets whether and how referrals to other servers are pursued. Valid values are: • `None` — Never chase the referred-to server. • `Subordinate` — Chase only referrals that are in a subordinate naming context in a directory tree. • `External` — (Default value.) Chase external referrals that are outside the directory tree. • `All` — chase referrals of either the subordinate or external type.

Attribute	Details
SearchScope	Optional. Sets the scope of the search. Available values: • Base — limits the search to the base object. Only one object is returned. • OneLevel — Search the immediate child objects of the base object, excluding the base object. • SubTree — (Default value.) Search the whole sub-tree, including the base object and all its child objects.
ServerPageTimeLimit	Reserved for future development. An integer value to set the number of seconds the server searches for an individual page result. The default value is (-1) which means to wait indefinitely. Paged searches are not yet supported by the Business Importer.
ServerTimeLimit	Optional. Sets the maximum number of seconds the server spends searching. Special value: • -1 — (Default value.) The server-determined default is enforced (for Active Directory, this is 120 seconds).
SizeLimit	Optional. An integer value to set the maximum number of objects the server returns in a search result. Special value: • 0 — (Default value.) The server-determined default is enforced (for Active Directory, this is 1000 entries).

Attributes for handling text files

Together, these attributes control how many lines from the start of a text file should be discarded in order to arrive at the data.

Attribute	Details
FileName	Required in order to skip any rows (if the next attribute is included). Sets the name of the text file in which to apply the RowsToSkip attribute, and should be identical to the one used in the Query attribute.
RowsToSkip	Optional. An integer value to set the number of rows to delete from the beginning of the text file identified with the FileName attribute.

Attributes for handling web service queries

Together, these attributes control the handling of SOAP requests.

Attribute	Details
SOAPHeaderValues	Optional. A string containing the values to include in the web service request header. The values must be formatted as name/value pairs, separated by a semi-colon, as follows: `SOAPHeaderValues="Name1=Value1;Name2=Value2…"`
SOAPXMLElement	Optional. A string containing the name of the element to be read in the web service response. If this attribute is not included, the data is read from the Body element (if it exists) or from the name of the web service method (if specified as such in the Query attribute, with the string “Result” added at the end).

ConnectionString Attribute

Each Import element has a ConnectionString attribute that defines an OLE-DB connection from the adapter to the external data source. However, the format of the ConnectionString attribute depends on the Type attribute of the Import element, which declares the type of data source to which the adapter connects for this particular import.

Note: *The Business Importer is a 32-bit application, and 32-bit OLE-DB connection strings must be used on 64-bit operating systems.*

Tip: *For guidelines about building and validating a connection string, see Validating Connection Strings.*

The following table shows, for each Type of data source that the import may use, the different format and attribute values for the ConnectionString.

Type attribute of Import	ConnectionString attribute of Import
ADSI	ADSI is used for directory data import, such as from Active Directory or eDirectory. Acceptable formats for ConnectionString when using this data source are: • If the Business Importer is currently running as an account known to the domain, the ConnectionString may be left empty (in which case, the current account credentials are used) • When authentication needs to be specified, use `Login=AcctName;Password=AcctPwd`

Type attribute of Import	ConnectionString attribute of Import

CSV	This import Type may be used both for files with delimited values (as in a true comma-separate value file), or for text files with fixed-length values.
	• For delimited columns, the format is:
	`Provider=Microsoft.Jet.OLEDB.4.0; Data Source=`*`FilePathOnly`*`;`
	`Extended Properties='text;HDR=Yes;FMT=Delimited'`
	where
	◦ *FilePathOnly* contains the path to the text file, but not the file name. (The file name is defined in the Query attribute of the Import element.)
	◦ HDR defines whether the first row (paragraph) of the file contains field/column names. When HDR=Yes, the first row is interpreted as the column names. When HDR=No, the field names are automatically assigned as F1, F2, ... F*n*.
	◦ FMT sets the delimiter. Valid values are:
	▪ Delimited (default) uses the Format setting taken from the following registry key:
	• For a 32-bit operating system:
	`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Jet\4.0\Engines\Text`
	• For a 64-bit operating system:
	`HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Jet\4.0\Engines\Text`
	Format may have any of the following values (same values for the connection string or in the registry).
	▪ Delimited(;) for values separated by semi-colons (;). Alternatively, any other character except the double quotation mark can be specified in the parentheses, including a space. This custom character becomes the separator.
	▪ CSVDelimited for standard comma-separated values files.
	▪ TabDelimited for values separated by the tab character.
	• For fixed length columns, the format is:
	`Provider=Microsoft.Jet.OLEDB.4.0;Data Source=`*`FilePathOnly`*`;`
	`Extended Properties='text;HDR=Yes;FMT=Fixed'`
	In all cases, to specify the properties of each column in the imported file, use a

`Type` attribute of `Import`	`ConnectionString` attribute of `Import`
	`schema.ini` file (see Editing a Schema.ini File).

`Excel`	The format depends on which version of spreadsheet files you are importing.
	• For `xls` files, the format is:
	`Provider=Microsoft.Jet.OLEDB.4.0;Data`
	`Source=`*`FullPathAndFileName`*`;`
	`Extended Properties=Excel 8.0;HDR=Yes`
	• For `xlsx`, `xlsb`, or `xlsm` files, the format is:
	`Provider=Microsoft.ACE.OLEDB.12.0;Data`
	`Source=`*`FullPathAndFileName`*`;`
	`Extended Properties=Excel 12.0 Xml;HDR=Yes;IMEX=1`
	Where, in both cases:
	• `HDR` defines whether the first row of the spreadsheet contains column names. When `HDR=Yes`, the first row is interpreted as the column names. When `HDR=No`, the column names are automatically assigned as `F1`, `F2`, ... `F`*`n`*.
	• `IMEX=1` is the safe way to retrieve data from mixed data columns.
	Refer to Working With Excel Files for additional information.

`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. This example is for a connection to an Excel file using a test DSN:
	`DSN=test;DriverId=790;FIL=excel`
	`8.0;MaxBufferSize=2048;PageTimeout=5;`

`OleDB`	OleDB is a generic driver that can be used with a variety of databases, such as Microsoft Access, Ingres, Paradox, and others. The prerequisite is that the corresponding OLE-DB driver has been installed and configured on the computer where the import is run. Example for Microsoft Access:
	`Provider=Microsoft.Jet.OLEDB.4.0;Data`
	`Source=`*`PathAndFilenameOf.mdbFile`*

Type attribute of `Import`	`ConnectionString` attribute of `Import`

`Oracle`	Oracle connections require the installation of an Oracle client provided by Oracle Corporation. The Oracle client should install the OLE-DB driver for Oracle. The format is:
	`Password=Password;User ID=Account;Data`
	`Source=OracleDataSourceName;`
	`Persist Security Info=True`
	Tip: *Do not include the provider in this connection string.*

`SqlServer`	The format depends on the authentication model.
	• For a database using Windows Authentication:
	`Integrated Security=SSPI;Persist Security Info=False;`
	`Initial Catalog=DatabaseName;Data Source=ServerName`
	Tip: *If the instance of SQL Server is not the default, append the instance name with a backslash separator:*
	`Data Source=ServerName\InstanceName`
	• For a database using SQL Server authentication:
	`Password=Password;Persist Security Info=True;User`
	`ID=Account;`
	`Initial Catalog=DatabaseName;Data Source=ServerName`
	Tip: *Do not include the provider in this connection string.*

`WebService`	Web services can be called using a SOAP request. The connection string must contain the URL and, if authentication is needed, the credentials:
	`URL=WebServiceURI`
	Or, with credentials:
	`URL=WebServiceURI; Login=Account;Password=Password`

`Type` attribute of `Import`	`ConnectionString` attribute of `Import`

`XML`	Allows for import of XML files. The connection string must contain the path and name of the XML file:
	`Path\FileName`
	Enclosing quotation marks are mandatory when the values include white space, and otherwise optional. Example:
	`"C:\temp\Data.xml"`

Query Attribute

Some drivers require special formats for the `Query` attribute of the `Import` element.

When the `Import` element is connecting to a database, the `Query` attribute must contain a valid SQL statement for the target database.

For other values of the import `Type`, the drivers may require a specific syntax. These special cases are shown in the table below.

`Type` (driver)	Query description
`ADSI`	ADSI queries follow the LDAP syntax for a search filter: • The string must be enclosed in parenthesis • Expressions can use the relation operators `<`, `<=`, `=`, `>=`, `>` and the compound operators `&` and `|`. For example, to return all objects of category ‘user’, and class ‘person’, with a non-blank email address: `(&(objectCategory=user)(objectClass=person)(mail=*))`

`CSV` (text file)	The syntax is:
	`select * from FileName`
	Example:
	`select * from Asset.csv`
	Do not include the path to the file in the query (the path is specified in the connection string for this `Type` of import).

Type (driver)	Query description

`Excel`	The default query (case insensitive) is:
	`select * from NameOfWorksheet`
	Example:
	`Select * from [Sheet1$]`
	In addition, Excel supports field enumeration and functions such `MID` or `ABS`.

`WebService`	The query may be either: • The name of the method to be called • The full SOAP request to the web service.
`XML`	No queries are supported.

Detailed Tracing

Highly detailed logging can be written to the database for individual imports. This is especially useful during development and testing.

In addition to the logging controlled by the `Log` element, you can set up detailed tracing for the operation of each `Import` element independently. This is done using the `TraceActions`, `TraceField`, and `TraceLifeTime` attributes of the `Import` element.

Three separate tables are populated in the database:

- To trace each import at a summary level, a record is created in the `ECMImportLog_Summary` table each time the Business Importer is started in import or simulation mode.
- To trace the objects included in the import, a record is created in the `ECMImportLog_Object` for each object included in the import (except for the `Custom` object).
- To track individual actions, a record is created in the `ECMImportLog_Detail` table for each action of the type(s) identified in the `TraceAction` attribute of the `Import` element.

***Tip:** Tracing can rapidly increase the number of records stored in the database. Be sure to set the `TraceLifeTime` attribute of the `Import` element to the minimum timespan consistent with your debugging needs.*

Details of each of the tracing records are listed in the tables below.

Tracing Imports (Summary)

A record is created in the `ECMImportLog_Summary` table each time the Business Importer is started in import or simulation mode. Each row contains the following information:

Column	Description
ImportID	Unique identifier for each record.
ImportName	The Name attribute of the import.
ImportType	The Type attribute of the import (SqlServer, Oracle, CSV, and so on).
Action	Import or Simulation.
StartDate	The date and time the import was started (database date and time).
EndDate	The date and time when the import finished (database date and time).
Status	Values: • 1 indicates success • 0 indicates that the Business Importer did not complete its task.
Processed	The number of records processed from the data source.
Rejected	The number of records rejected from the data source. The same record can be rejected for multiple reasons, but will be counted only once.

Tracing Objects

A record is created in the ECMImportLog_Object for each object included in the import (except for the Custom object). Each row contains the following information:

Column	Description
ImportObjectID	Unique identifier for each record.
ImportID	Reference to the ECMImportLog_Summary table.
ObjectName	The name of the object as specified in the XML.
ObjectType	The type of the object.
StartDate	The date and time the import of this object was started (database date and time).
EndDate	The date and time when the import of this object finished (database date and time).
Processed	The number of records for this object processed from the data source.
Rejected	The number of records for this object rejected from the data source. The same record can be rejected for multiple reasons, but will be counted only once.
Matched	The number of records for this object matched between the data source and the compliance database.
Updated	The total number of records for this object updated in the compliance database.
Created	The total number of records for this object created in the compliance database.

Column	Description
Deleted	The total number of records for this object deleted from the compliance database.

Tracing Records

A record is created in the ECMImportLog_Detail table for each action of the type(s) identified in the TraceAction attribute of the Import element. The TraceAction attribute specifies whether the creation, updating, rejection, or deletion of individual records needs to be tracked in the database logging.

Note: *A limited set of database entities are not tracked in the ECMImportLog_Detail table. There are primarily relationships between objects. For example, the license allocation object cannot be logged in this way.*

Each row contains the following information:

Column	Description
ImportDetailID	Unique identifier for each record.
ImportID	Reference to the ECMImportLog_Summary table.
RecordNumber	ID of the record considered.
Action	Creation, Deletion, Update, or Rejected.
MGSRecordKey	ID of the record in the FlexNet Manager Suite compliance database.
ImportObjectID	Reference to the ECMImportLog_Object table.
RecordDescription	To more clearly identify the individual record, this column is populated as defined by the TraceField attribute of the Import element. For more information, see the TraceField attribute description in Import Element.
Message	This column contains the reasons for discarding records.

Object Element

The Object element defines one object in the central compliance database that is created or updated by an Import.

Every Import element contains at least one, and possibly several, Object elements. In turn, an Object element contains many Property elements that map the incoming data values to the properties of the database object within the compliance database of FlexNet Manager Suite.

The ordering of the Object elements is important, as they are processed in the same order that they appear in the adapter XML file. Therefore (for example), if an adapter imports purchases that reference new vendors, it is important that the Object element for vendors appears earlier in its parent Import list than the Object element for the purchases. If this ordering is wrong, the overall processing may fail when a purchase (processed too early) fails to reference the vendor that has not yet been created.

With multiple Object elements in an import, it is critical that their Name attributes are unique. Other than the requirement for uniqueness, the Name is an arbitrary value to assist your understanding of the incoming data. The mapping to the destination database objects is done by the mandatory Type attribute.

Example

```
<Object
   Name="MachineRoomAsset1"
   Type="Asset"
   Output="RefAssetID"
   Create="true"
   Update="true"
   UpdateRule="AddToExistingRecords"
   CustomComputerMatching=""
   <Property...>
     ...
   </Property> ...
</Object>
```

Contains

Child	Comments
Property	Mandatory. Identifies a single property of an object in the central compliance database, maps a data source column to this property, and specifies appropriate business rules governing the data.

Contained by

Import.

Supported attributes

Attributes of the Object element are divided into:

- Those applying in general to all objects
- Additional attributes specific to the import of custom objects.

General purpose attributes

These attributes apply to all Object elements.

Attribute	Details
Name	Mandatory. Uniquely identify the imported object in the XML file. Special characters and spaces should not be used.
Type	Mandatory. Specifies the compliance database object affected by the import. The value must be an exact match for one of the values listed in Object Type Attribute.

Attribute	Details
`Output`	Mandatory. Specify a column name (which must be unique across *all* objects within the current `Import`) which is added to the object definition to include the ID of each record of this `Object` that is created, updated, or looked up in the central operations databases (specifically, the). When objects need to reference one another, they use the ID contained in this variable.
`Create`	Mandatory. Boolean. Where the incoming data does not match an existing database object of the declared `Type`: • `true` means that a new record of the same type is created • `false` means that the incoming record is rejected as unmatched.
`Update`	Mandatory. Boolean. Where the incoming data matches an existing database object of the declared `Type`: • `true` means the existing record is updated with the incoming values • `false` means that the incoming record is rejected as already existing.
`UpdateRule`	Optional. Sets the behavior for updating relationships between this object and other related objects in the compliance database (based on matching key property values). Rules are specific to each relationship, and are listed in UpdateRule Attribute. Only one update rule may be specified for each object.

`CustomComputer Matching`	Mandatory if the import object is an asset (and otherwise ignored). This attribute allows you to disable or replace the logic that links newly-created assets with existing inventory devices already in the compliance database. Available values:
	• An empty string means that the built-in logic will be executed:
	`CustomComputerMatching=""`
	• A string of only spaces means that the built-in logic will be disabled:
	`CustomComputerMatching="   "`
	• You may provide the name of a stored procedure, or a raw SQL statement, that will link newly-created assets with existing inventory devices. For further information and examples, see Customizing the Asset-Device Linkage.

Attributes specific to custom objects

These additional attributes should also be considered when you are directly manipulating the compliance database.

Attribute	Details
QueryType	Optional. Specifies how the central compliance database should be updated. Possible values are: • Text (the default) means that the Query property contains a raw SQL statement to execute. • StoredProcedure means that the Query property contains the name of a stored procedure to be executed on the central compliance database.
Query	Mandatory (for custom objects). Depending on the setting for QueryType, the Query contains either: • A raw SQL statement to execute • The name of a stored procedure to be executed on the central compliance database.
Timeout	Mandatory (for custom objects). An integer value to set the timeout in seconds, when running a custom query.
SendLogIdAsParam	Optional. Supported only for on-premises implementations, and only for adapters that are running on the central application server. Applicable only when QueryType="StoredProcedure". Boolean: • false (the default) means that no log ID is passed as a parameter to the stored procedure. • true allows your stored procedure to log details under the current log ID. When SendLogIdAsParam=true, a @FinImporterLogID parameter in your stored procedure is given the value of the current log ID in the FinImporterLog table of the database (which is automatically maintained by the Business Importer, and not described in the standard product schema).

Object Type Attribute

The database objects accessible through the Business Importer.

Each Import element includes one or more Object elements for import into the central compliance database. Each Object element is related to a database object through its Type attribute.

***Tip:** In several cases, an imported Object may insert new values into the related database table (identified through its Type attribute). However, for the Type values below that are marked with an asterisk (*), inserting new values is only supported for single tenant on-premises implementations; and is not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.*

The Type attribute must exactly match one of the following values:

- AcquisitionMode*

- `Asset`
- `AssetComplianceStatus*`
- `AssetStatus*`
- `AssetType*`

Note: For presentation in the web interface of FlexNet Manager Suite (on-premises implementations only), new asset types inherit their XML template screen definitions from the `Workstation` asset type. They do not require a link to an inventory device (computer); but this requirement can be enforced by setting the `managed` property to `1`.

- `AssetWarrantyType`
- `Category`
- `ChargeBackPeriodType*`
- `ComplianceComputerConnection`
- `ComplianceDomain`
- `ComplianceResponsibility`
- `Computer`
- `ComputerChassisType*`
- `ComputerInventorySourceType*`
- `ComputerRole*`
- `ComputerStatus*`
- `ComputerType`

***Note:** No new computer types can be created in the central compliance database using the Business Importer.*

- `Contract`
- `ContractAsset`
- `ContractLicense`
- `ContractPeriodType*`
- `ContractStatus*`
- `ContractType*`

***Note:** New contract types (on-premises implementations only) inherit their XML template screen definitions from the `General` contract type.*

- `CorporateUnit`
- `CostCenter`
- `Currency`
- `CurrencyRate`
- `CurrencyRateSnapshot`
- `Custom` (for details, see Custom Objects)
- `DepreciationMethod*`
- `Document`
- `EndOfLifeReason`
- `LeaseEndReason*`
- `LeasePeriodType*`
- `License`
- `LicenseComplianceStatus`
- `LicenseDuration*`
- `LicenseStatus*`
- `LicenseType*`

__Note:__ New software license types (on-premises implementations only) inherit their XML template screen definitions from the `Enterprise` software license type. By default, new software license types are treated as the `Volume` license type.

- `LicenseWarrantyType*`
- `Location`
- `PaymentSchedule`
- `PaymentScheduleAsset`
- `PaymentScheduleCategory*`
- `PaymentScheduleDetail`
- `PaymentScheduleDetailStatus*`
- `PaymentScheduleLicense`
- `PaymentScheduleTerm*`
- `PaymentScheduleType*`

- `PeriodType*`
- `Publisher`
- `PuchaseOrderType*`
- `PurchaseOrder`
- `PurchaseOrderDetailType*`
- `PurchaseOrderLine`
- `PurchaseOrderLineAsset`
- `PurchaseOrderLineLicense`
- `PurchaseOrderStatus*`
- `ResponsabilityType*`
- `ShippingMethod*`
- `SoftwareLicenseAllocation`
- `SoftwareLicenseAllocationStatus*`
- `SoftwareLicenseKey`
- `SoftwareLicenseKeyType*`
- `TermAndCondition`
- `TermAndConditionType*`
- `User`
- `UserStatus`
- `UserSuffix`
- `UserTitle`
- `Vendor`

UpdateRule Attribute

The update rule controls how relationships with other objects are managed during the import.

As well as the basic objects in the central compliance database, there are many relationships between them. The Business Importer allows you to manage what happens to the links between database objects, especially when objects referenced by the links are no longer present in the incoming data from the data source. For example, suppose that a user disappears from a regularly-imported list of employees. Should the system check whether that user was linked to any contract records in a particular role, and if so remove the link?

Two settings apply to all object relationships, and the remaining values are variants applying to specific relationships.

Use the values appropriate to the object you are importing. For example, suppose your import includes license objects and contract objects, and you wish to remove links (previously imported) when instances of either object are not matched in either the imported data or the existing database records:

- For the license object, set `UpdateRule="RemoveExtraContractsFromLicenses"`
- On the contract object, set `UpdateRule="RemoveExtraLicensesFromContracts"`.

In the following tables, any value wrapped for documentation should be entered as a single unbroken string.

Common to all relationships

`UpdateRule` value	Description
`AddToExistingRecords`	This setting means that the import never deletes any existing relationships. New links between objects are added where required and permitted. ***Note:** This is the default behavior when the `UpdateRule` is omitted.*
`RejectDuplicateRecord`	Normally, duplicate records in the source data are ignored (that is, the first instance is processed, and matching subsequent instances are ignored). This value of the `UpdateRule` changes behavior so that, rather than being ignored, duplicate records are rejected. As a result, they are logged. You can use the logs to analyze and remove the duplicates at the source.

Contract - Asset relationships

`UpdateRule` value	Description
`RemoveExtraAssetsFromContracts`	Removes asset links from contracts where assets were not found in the incoming data.
`RemoveExtraContractsFromAssets`	Removes contract links from assets where contracts were not found in the incoming data.
`RemoveExtraLinks AssetsContracts`	For contracts and assets in the incoming data (from this adapter only), this setting clears any recorded links that are not also included in the incoming data (that is, repeated since the previous import from this source).

Contract - License relationships

`UpdateRule` value	Description
`RemoveExtraLicenses FromContracts`	Removes license links from contracts where licenses were not found in the incoming data.

`UpdateRule` value	Description
`RemoveExtraContracts FromLicenses`	Removes contract links from licenses where contracts were not found in the incoming data.
`RemoveExtraLinks LicensesContracts`	For contracts and licenses in the incoming data, this setting clears all links that are not also included in the incoming data.

Purchase - License relationships

While the web interface for FlexNet Manager Suite focuses on individual purchase records, the terminology used in the Business Importer tends to focus more on the purchase orders that contain one or more purchases. Here, each individual purchase is often referred to as a 'purchase order line'.

`UpdateRule` value	Description
`RemoveExtraLicenses FromPurchaseOrders`	Removes license links from purchases, where licenses were not found in the incoming data.
`RemoveExtraPurchaseOrders FromLicenses`	Removes purchase links from licenses, where purchases were not found in the incoming data.
`RemoveExtraLinks PurchaseOrdersLicenses`	For purchases and licenses in the incoming data, this setting clears all links that are not also included in the incoming data.

Purchase - Asset relationships

While the web interface for FlexNet Manager Suite focuses on individual purchase records, the terminology used in the Business Importer tends to focus more on the purchase orders that contain one or more purchases. Here, each individual purchase is often referred to as a 'purchase order line'.

`UpdateRule` value	Description
`RemoveExtraAssetsFrom PurchaseOrders`	Removes asset links from purchases, where assets were not found in the incoming data.
`RemoveExtraPurchaseOrders FromAssets`	Removes purchase links from assets, where purchases were not found in the incoming data.
`RemoveExtraLinks PurchaseOrdersAssets`	For purchases and assets in the incoming data, this setting clears all links that are not also included in the incoming data.

Payment schedule - Asset relationships

`UpdateRule` value	Description
`RemoveExtraAssetsFrom PaymentSchedules`	Removes asset links from payment schedules, where assets were not found in the incoming data.
`RemoveExtraPaymentSchedules FromAssets`	Removes payment schedule links from assets, where payment schedules were not found in the incoming data.

UpdateRule value	Description
RemoveExtraLinks PaymentSchedulesAssets	For payment schedules and assets in the incoming data, this setting clears all links that are not also included in the incoming data.

Payment schedule - License relationships

UpdateRule value	Description
RemoveExtraLicenses FromPaymentSchedules	Removes license links from payment schedules where licenses were not found in the incoming data.
RemoveExtraPaymentSchedules FromLicenses	Removes payment schedule links from licenses where payment schedules were not found in the incoming data.
RemoveExtraLinks PaymentSchedulesLicenses	For payment schedules and licenses in the incoming data, this setting clears all links that are not also included in the incoming data.

User - Contract relationships

UpdateRule value	Description
RemoveExtraUsers FromContracts	Removes user links from contracts where users were not found in the incoming data.
RemoveExtraContracts FromUsers	Removes contract links from users, where contracts were not found in the incoming data.
RemoveExtraLinks ContractsUsers	For users and contracts in the incoming data, this setting clears all links that are not also included in the incoming data.

License allocation - Computer relationships

Individual license entitlements can be allocated to specific inventory devices (computers).

UpdateRule value	Description
RemoveExtraLicenseAllocation FromComputer	Removes license allocations links from inventory devices, where licenses were not found in the incoming data.
RemoveExtraComputerFrom LicenseAllocation	Removes inventory device links from license allocations, where the inventory devices were not found in the incoming data.
RemoveExtraLink ComputerLicenseAllocation	For licenses and inventory devices in the incoming data, this setting clears all links that are not also included in the incoming data.

Custom Objects

Within an Import, you can include an Object of Type="Custom". These custom objects are used to run SQL stored procedures, or raw SQL statements, against the central compliance database when customizations are needed. Using either method, the SQL statements are exercised within the same transaction as the data import. The choice between using a stored procedure or a direct SQL statement is set by the QueryType attribute:

```
QueryType="StoredProcedure"
```

or

```
QueryType="Text"
```

This example shows how to exercise a stored procedure as part of your import transaction:

```
<Object
   Name="ClearGroupExComments"
   Type="Custom"
   QueryType="StoredProcedure"
   Query="MyStoreProcedure">
</Object>
```

Custom objects are not automatically logged as part of the detailed tracing available in the database log (see Detailed Tracing). However, you can provide your own logging, using the one special keyword supported within the SQL:

Keyword	Description
[LOG_IMPORT_ID]	Returns the current value of the ImportID column in the ECMImportLog_Summary table.

The following example shows writing a log entry into the ECMImportLog_Object table, providing a foreign key into the ECMImportLog_Summary table. For details about these tables, see Detailed Tracing.

```
<Object
   Name="ClearGroupExComments"
   Type="Custom"
   Query="Insert into ECMImportLog_Object([ImportID], [ObjectName],
          [StartDate], [EndDate], [ObjectType], [Status])
          values ([LOG_IMPORT_ID], 'My Custom Object',
          getdate(), getdate(),'Custom', 1)">
</Object>
```

Localization of New Values

Most static string values in the compliance database support translation. These static values usually populate enumerated lists, such as a list of asset types, or the possible values of asset status.

The translation mechanism uses three database tables:

- ComplianceResourceString gives each string using a unique string ID, and provides the default content for the string if no localized versions are found.

- `ComplianceCultureType` lists the available culture types for which localizations are available.
- `ResourceStringCultureType` stores all required translations of a string identified by a string ID and the culture type.

When your `Import` defined for an `Object` requires new values in such lists (for example, creating a new asset type or a new asset status), a unique string ID is automatically created by the Business Importer and the base record is created in the `ComplianceResourceString` table, using the imported new value as the default.

In order that anyone using the web interface for FlexNet Manager Suite in another language is not left with a blank where the new value should be, the same value is also written into all available culture types as the 'translated' values within each culture. This is simply a placeholder, and is unlikely to be the final desired value for each of your supported culture types. If you need to display a localized value for your new static string in the web interface of FlexNet Manager Suite, ensure that the appropriate localized version(s) are also edited in the `ResourceStringCultureType` table. For example:

- Suppose you import a new asset status value of `Under repair`.
- Immediately after import, this value is displayed in all cultures. An operator using a French presentation currently sees `Under repair`.
- Using SQL Server Management Studio, you open `ResourceStringCultureType` and edit the corresponding string for `fr-FR` from `Under repair` to `En réparation`.
- From this edit onwards, an operator using a French presentation sees `En réparation`.

***Note:** When searching the database for existing records, the match is performed using the translation(s) found in the `ResourceStringCultureType` table.*

Data Synchronizing Rules

Within FlexNet Manager Suite, there are business rules that can be set to manage data updates across related objects. The Business Importer honors the settings of the following rules, as at the time of each import. The business rules are available on various tabs in the **Systems Settings** page of the web interface for FlexNet Manager Suite:

- Synchronize enterprise groups in device and asset properties (**Assets** tab)
- Synchronize enterprise groups in asset and sub asset properties (**Assets** tab)
- Synchronize enterprise groups in user and device properties (**Inventory** tab).

Property Element

For each database object, there are a number of properties; and these elements map the incoming details to the correct destination properties.

Each `Property` element maps a column of source data to a single property (or column) of an object in the central compliance database for FlexNet Manager Suite. Each `Property` also contains business rules to manage the processing needed to convert from the data source to the destination database.

Example

```
<Property
        Name="Serial Number"
        Type="serialnumber"
        Update="Always"
        Value="SerialNo"
        ValueType="Field Value"
        UseForMatching="True">
</Property>
```

Contains

No child elements: `Property` is an empty XML element, using only its attributes.

Contained by

`Object`.

Supported attributes

Attribute	Details
`DataType`	Optional, but required when `IsCustomField=true`. It declares the data type of the custom property so that the data is correctly parsed. Valid values are: • `int` • `date` • `numeric` • `boolean` • `string` (default). ***Tip:*** *For standard (non-custom) properties, the data type, format, and precision for each destination property is automatically read from the compliance database schema.*

Attribute	Details

`FieldMask`	Identifies any constant prefix and/or suffix on values in the incoming property records that do not exist in the target database. When the Business Importer assesses incoming records for matches in the target database, it first augments the target key value with the additional data in the field mask, and then assesses the match.
	The formats for declaring this value are:
	`FieldMask="'prefix'+ [targetColumnName] + 'suffix'"`
	`FieldMask="'prefix'+ [targetColumnName]"`
	`FieldMask="[targetColumnName] + 'suffix'"`
	Notes:
	• In general, the value replacing *`targetColumnName`* will be the same as the value of the `Type` attribute — the name of the target column in the target database.
	• This column name must be enclosed in literal square brackets, indicating "value of".
	• The *`prefix`* and *`suffix`* placeholders can be replaced with any text string.
	• The plus sign (+) is required for string concatenation. White space around this operator is optional.
	For example, suppose your source data includes location names with a constant prefix `Locn-`, so that the source data represents Australia as `Locn-Australia`. However, in the target central operations databases, the locations do not have this superfluous prefix, and there the location is simply called `Australia`. Without some intervention, the source data could never update records in the target database, because the values are not matched. To solve this, declare:
	`FieldMask="'Locn-'+[groupcn]"`
	This mask concatenates the fixed prefix with each value of the `groupcn` column in the target database, and the combination is tested against the incoming data. Therefore, the incoming record `Locn-Australia` is matched by the evaluated `FieldMask` of `Locn-Australia`, and the Business Importer now knows to update the entry for the `Australia` location in the target database.

Attribute	Details
`Format`	Declares the format of date/time values (when `DataType="date"`) and numeric values (when `DataType="numeric"`). These are formats identified for the source data, allowing the Business Importer to convert to the format required for storing in the target database. The following references may assist: • For standard date/time formats: https://msdn.microsoft.com/en-us/library/az4se3k1.aspx • For custom date/time formats: https://msdn.microsoft.com/en-us/library/8kb3ddd4.aspx • Decimal number formats: https://msdn.microsoft.com/en-us/library/d8ztz0sa%28v=vs.110%29.aspx • Integer formats: https://msdn.microsoft.com/en-us/library/8wch342y%28v=vs.110%29.aspx
`IsCustomField`	Optional. Boolean: • `true` means that this property is a custom property added to the compliance database • `false` (the default) means that the property is a standard one documented in the FlexNet Manager Suite Schema Reference PDF file. See also the `DataType` attribute. For more details on these attributes, see Custom Properties. ***Note:*** *The automatic creation of custom properties defined in the adapter XML file is supported only for on-premises implementations when the adapter is running on the central application server (and not supported on inventory beacons where the adapter runs in disconnected mode). However, once a custom property exists in the operations databases, any adapter can add values to the custom property.* Custom properties are only available for the following objects in the compliance database: • Application • Asset • Computer • Contract • Purchases (Purchase Order header and lines) • Software License • User. For more about customization, see the FlexNet Manager Suite System Reference PDF.

<table>
<tr><th>Attribute</th><th>Details</th></tr>
<tr><td><code>MatchingMask</code></td><td>
<p>Supported only for on-premises implementations where the adapter is running on the central application server (and specifically not permitted for adapters running on inventory beacons). Except when dealing with properties of enterprise groups, this is required when <code>MatchingMode="like"</code>, for which it declares the pattern used to find matching records. The value of the property in the incoming data stream is represented as the special placeholder <code>[value]</code> (with the square brackets).</p>
<p>The <code>MatchingMask</code> attribute uses standard SQL syntax to declare a pattern to match. The exact syntax varies, depending on the value of the <code>ValueType</code> property. For example, to match a record that ‘contains’ the incoming value (with other possible content), the two syntax examples are:</p>
<ul>
<li>For <code>ValueType="Fixed Value"</code>:
<pre>MatchingMask="%[value]%"</pre></li>
<li>For <code>ValueType="SQL Value"</code> or <code>ValueType="Field Value"</code>:
<pre>MatchingMask="'%' + [value] + '%'"</pre></li>
</ul>
<p>Example: The following XML looks up assets (in the compliance database) by serial number based on a partial match:</p>
<pre>
<ObjectName="asset" Type="asset" Output="assetoutid">
   <Property
       Type="SerialNo"
       Name="SerialNo"
       Value="SerialNumber"
       ValueType="FieldValue"
       UseForMatching="True"
       MatchingMask="%[value]%"
       MatchingMode ="like"/>
   …
</Object>
</pre>
</td></tr>
<tr><td><code>MatchingMode</code></td><td>
<p>Optional. Specifies how this property will be matched against the existing data in the central compliance database. Available values:</p>
<ul>
<li>= (the equals sign) (default). Matches must be exact. (<code>MatchingMask</code> is ignored.)</li>
<li><code>like</code> allows the use of a matching mask, declared in the <code>MatchingMask</code> attribute.</li>
</ul>
<p><b><i>Tip:</i></b> <i>The <code>MatchingMask</code> attribute is not required in the special case of properties of enterprise groups.</i></p>
</td></tr>
<tr><td><code>Name</code></td><td>Mandatory. This is the friendly name of the property. The attribute is exclusively used in the adapter XML file, and need not match either incoming or target columns names (for which see <code>Value</code> and <code>Type</code> respectively).</td></tr>
</table>

<table>
<tr><th>Attribute</th><th>Details</th></tr>
<tr><td>

`OnMissingFieldValue`

</td><td>

Optional. This attribute dictates the behavior of the Business Importer when this clash occurs:

- The adapter XML file defines an incoming column name (declared in `Value` when `ValueType="FieldValue"`)
- The source data does not include a column of the same name.

Support values are:

- Empty string, not declared, or given any value other than the following two: TheBusiness Importer raises an error to highlight that you have mapped a source column that does not exist to a target column.
- `Discard Property` means that records for the incoming instances of the object are created and updated, but the missing property is silently ignored.
- `Discard Object` means that no instances of the object are created in the target database.

For example, suppose you have the following `Property` on a location `Object` in your adapter XML file:

```
<Property
    Name="Name"
    Type="groupcn"
    OnMissingFieldValue="Discard Object"
    Update="Never"
    Value="Loaction"
    ValueType="FieldValue"
    UseForMatching="True"
    RegExSplit="/" />
```

The combination of `ValueType="FieldValue"` and `Value="Loaction"` tell the Business Importer to map input from the `Loaction` column into the target database. However, if the source data does not share the same typographical error, and instead has a column called `Location`, the proposed column name cannot be found. Now `OnMissingFieldValue="Discard Object"` means that no locations will be imported, because the entire object is discarded based on the failure to find the key field. The fix, of course, is to correct the error in the adpater XML file.

</td></tr>
</table>

Attribute	Details
`RegexOption`	Optional. Specifies the regex options applying to the `RegexString` attribute. Supported values are: • `Cultureinvariant` specifies that cultural differences in matching methods are ignored. • `Ecmascript` specifies that ECMAScript compliant behavior is enabled for the expression. (ECMAScript is Javascript compliant with the ECMA-262 standard.) • `Ignorecase` specifies case-insensitive matching. • `Ignorepatternwhitespace` specifies that unescaped white space is excluded from the pattern. • `Multiline` specifies multi-line mode. • `Rightoleft` specifies that the search moves from right to left instead of left to right. • `Singleline` specifies single line mode. • `None` (default) means that no options are used.

`RegexOrder`	Optional. May only be used with the `GroupCN` property of enterprise groups; and is used in conjunction with the `RegexSplit` attribute. This `RegexOrder` attribute declares whether the compound string of enterprise groups reads from parent down to child (leaf node last), or from child up to parent (leaf node first). It supports the following values:
	• `Regular` (default) means the parent-to-child values are read from left to right. A `GroupCN` value ordered this way might be:
	`USA/Boston/100 North Washington`
	• `Reverse` means the values are read leaf node first, with parents to the right. A `GroupCN` value ordered this way might be:
	`100 North Washington/Boston/USA`

Attribute	Details
`RegexReplace`	Optional. Specifies one or more characters or strings in the incoming values for the current `Property` that are to be replaced by the corresponding value(s) declared in the `RegexReplaceBy` attribute. (The `RegexReplace` and `RegexReplaceBy` attributes must be considered as a pair.) • To specify multiple strings to replace, separate them with either a comma (`,`) or hash (#) character. • The number of separate strings in this `RegexReplace` attribute must match the number in the `RegexReplaceBy` attribute (any extra values in either one are ignored). • Spaces are significant, and are included in the pattern matching. (Don't leave extra spaces around the separator characters.) • With multiple strings, order is correlated between these two attributes (the first target string becomes the first replacement, and so on). Example: `RegexReplace = "Microsoft Corp.,Microsoft Corporation,Adobe Inc."`
`RegexReplaceBy`	Optional, but required when `RegexReplace` is used. Specifies one or more characters or strings used to replace the corresponding value(s) declared in the `RegexReplace` attribute. (The `RegexReplace` and `RegexReplaceBy` attributes must be considered as a pair.) • To specify multiple replacement strings, separate them with either a comma (`,`) or hash (#) character. • The number of separate strings in this `RegexReplaceBy` attribute must match the number in the `RegexReplace` attribute (any extra values in either one are ignored). • Spaces are significant, and are included in the replacement text. (Don't leave extra spaces around the separator characters.) • With multiple strings, order is correlated between these two attributes (the first replacement substitutes for the first target string, and so on). Example: `RegexReplaceBy = "Microsoft,Microsoft,Adobe"` When you combine the examples given for both attributes in the processing of this `Property` (such as the name of a vendor), all instances of either “Microsoft Corp.” or “Microsoft Corporation” are standardized to “Microsoft”, and all instances of “Adobe Inc.” are standardized to “Adobe”.

<table>
<tr><th>Attribute</th><th>Details</th></tr>
<tr><td>RegexSplit</td><td>
<p>Optional. Specifies the character on which to split the incoming property into separate values. May only be used with the GroupCN property of enterprise groups. The GroupCN values may contain an entire path through the tree of enterprise groups in a single column (imported as one Property). For example, a location may be specified as the following path:</p>
<pre>
USA/Boston/100 North Washington
</pre>
<p>The goal is to spit this path into separate location entities, organized in parent/child relationships to form a tree. The RegexSplit attribute specifies the character separator (in this example, the forward slash) used to split the GroupCN values into multiple database entities.</p>
<p>The above example value can be entered as separate locations in the compliance database, with the correct parent-child linkages, using the following XML:</p>
<pre>
<Object Name="Location" Type="Location"
        Output="locationoutid" Create="True" >
    <Property
        Type="GroupCN"
        Name="Name"
        Update="Never"
        Value="LocationFullPath"
        ValueType="FieldValue"
        UseForMatching="True"
        Regexsplit="/" />
    <Property
        Type="GroupExID"
        Name="Parent ID"
        Update="Never"
        Value="locationoutid"
        ValueType="FieldValue"
        UseForMatching="True"
        UseNullValueForMatching="removeproperty"/>
</Object>
</pre>
</td></tr>
<tr><td>RegexString</td><td>Optional. Sets the regular expression applied to incoming values of this property. Matched data is extracted and included in the import, while those parts of values that were unmatched are discarded. If there is no match found in an incoming value of this property, the output is am empty string. Processing of the regular expression may be modified by the value of RegexOption.</td></tr>
</table>

Attribute	Details
`Type`	Mandatory. The type uniquely identifies the property within the object in the compliance database, and must be an exact match for the name of an existing database column. Details of the compliance database schema are available in the *FlexNet Manager Suite Schema Reference* PDF file. Custom properties are supported on relevant objects (see `IsCustomField`). ***Important:*** *Do not set the* `Type` *attribute equal to the database ID of the enclosing object. (For example, if this is a property of an* `Asset` *object, do not set the* `Type` *attribute to* `AssetID`*.) Database IDs cannot be directly referenced, set, or updated by the Business Importer. Pointing this attribute to a database ID will produce erratic results and likely crashes of the Business Importer.*

Attribute	Details

`Update`	Optional. Sets the business rule for updating the property if there is already a value in the central compliance database. Possible settings:
	• `Always` (default) means that any new incoming value always overwrites the existing value. This includes incoming blanks or nulls removing pre-existing values in the compliance database. (To prevent the latter, use `Do Not Blank` instead.)
	`[Any new value]  >>  [Any existing value]  =  [New value]`
	• `Never` means any existing value in the compliance database is preserved, and any new incoming value is ignored. This might be used, for example, when you only want to look up a property, without updating it.
	`[Any new value]  >>  [Any existing value]  =  [Existing value]`
	• `If Blank` means that value is updated only when the operations databases has this column blank (null value). However, if the database already contains a value here, it is preserved, and the incoming value ignored.
	`[Any new value]  >>  [Any existing value]  =  [Existing value]`
	`[Any new value]  >>  [Blank value]         =  [New value]`
	• `Do Not Blank` means the property in the target database will be updated except when the incoming value is empty. An incoming null leaves any existing value in the compliance database intact.
	`[Any new value]  >>  [Any existing value]  =  [New value]`
	`[Blank value]    >>  [Any existing value]  =  [Existing value]`
	Note: *Most properties in the supplied adapter XML files (for both disconnected mode on inventory beacons, and for connected mode on the central application server of an on-premises implementation) are set to* `Do Not Blank` *to prevent accidental removal of existing data when the adapter is used. The exceptions to this general principle are:*
	• *The* `Property` *element has* `UseForMatching="true"`*, or*
	• *The parent* `Object` *element has* `Create="false"` *and* `Update="false"`*.*

Attribute	Details
`UseForMatching`	Optional. Boolean: • `false` (default) means this property plays no part in matching incoming data with existing records in the compliance database. • `true` means this property (perhaps in conjunction with others) acts as a key for retrieving existing records.
`UseNullValueForMatching`	Optional. Sets the business rule to apply when the value is empty or null and `UseForMatching=true`. Note that this is not a Boolean. Possible values are: • `True` means the null value will be used for matching, and corresponding records with a null or empty value for this property will be returned. • `False` (default) means the incoming record containing the empty or null value is rejected. • `Ignore` means the lookup is performed without this property. This requires the use of multiple properties for matching, and the values of the other matching keys are used. • `RemoveProperty` is a synonym for `Ignore`.
`Value`	Mandatory. Defines the value of the property to be written into the compliance database. Depending on the value of the `ValueType` attribute, the `Value` may be: • A literal value • The name of the column in the source data that includes the appropriate value • The `Output` attribute from a preceding `Object` in the current `Import` • A fragment of SQL (optionally including columns from the data source) that defines the value. **Note:** *If `ValueType="Field Value"` and the `Value` attribute is empty, the property is ignored during the import.*

Attribute	Details
`ValueType`	Optional. Determines how the `Value` attribute for the `Property` should be interpreted: • `Field Value` (default): in this case the `Value` attribute contains either of: ◦ The name of a column from the initial query (either a column name in a CSV or XLSX source file, or if the data is a result of an SQL query, the SQL column name) ◦ The value of the `Output` attribute of one of the preceding `Object` elements in the current `Import`. For example, suppose that an update to a location `Object` had an `Output` attribute set to `locationoutid`, which saves the ID of the affected record. This can now be written as a `Value` for a `Property` on a different `Object`, creating a foreign key to that location record. • `Fixed Value`: the `Value` attribute contains the value itself. This may be a value-by-reference, where the `Value` attribute specifies a reserved name in square brackets, such as `[TODAY]` (which inserts the current date/time), or `[ROWNUMBER]` (which inserts the number of the current row in the incoming data stream). • `SQL Value`: the `Value` attribute contains an SQL expression that, when run on the source database, returns the value. This fragment may include a column from the data source.

Custom Properties

Two attributes of `Property` are critical to the creation and use of custom properties through the Business Importer:

- `IsCustomField=true` is required.
- The `Type` attribute contains the name or description of the custom property. When adding new values to a custom property previously defined, the `ComplianceCultureType` table is searched for a matching value for the `Type` attribute (a match then identifies the database storage for the custom property). This search is initially conducted according to the local culture setting on the central application server. If there is no match found in the localized strings for that culture, a search is made using the default culture (en-US). If no match for the `Type` value is found, the custom property is new, and is automatically created in the compliance database. (For more details about localization tables, see Localization of New Values.)

Note: *An attribute `DataType` is now deprecated, as custom property values are stored internally as strings in the operations databases.*

Custom properties are only available for the following objects in the compliance database:

- Application
- Asset
- Computer

- Contract
- Purchases (Purchase Order header and lines)
- Software License
- User.

For example, suppose that the incoming data stream includes a `WarrantyStartDate` column, intended to be a custom property of an Asset object in the compliance database. There, the custom property is to be called `AssetStartOfWarrantyDate`. Here is an example definition of this property in the XML:

```
<Property
    Name="CustomAssetStartOfWarrantyDate"
    IsCustomField="true"
    Type="AssetStartOfWarrantyDate"
    Update="IfBlank"
    Value="WarrantyStartDate"
    ValueType="Field Value"
    UseForMatching="False" />
```

Log Element

The log element is used to create log files tracing the activity of the Business Importer. This element is available by default for a business adapter running in connected mode on your central application server, and can be manually added for adapters running in disconnected mode on an inventory beacon. When added in disconnected mode (perhaps using the Business Adapter Studio to specify the logging), the log contains only query-to-query information, and naturally excludes any data about server-side database actions.

Example

This example creates a log, using the date, time, and import name for the file name:

```
<Log
    Name ="NewLog"
    Output="file"
    Loglevel="warnings"
    filename="[DATE][TIME][IMPORT NAME].log.txt">
</Log>
```

Contains

No further elements: `<Log />` is an empty XML element, relying entirely on its attributes.

Contained by

`Import`.

Supported attributes

Attribute	Details
`Name`	Mandatory. This is the friendly name of the log element.
`Output`	Mandatory. Sets the destination of the log. Possible values are: • `Console` means send the log output to the console. • `File` means save the log to a file on the disk (see `FileName` attribute).
`Content`	Mandatory. Sets the content of the log. Valid values are: • `Detailed` means information for each object is logged according to the `LogLevel` specified. • `Summary` means a summary for each object is output at the end of the import. • `All` provides a combination of the `Summary` and `Detailed` log.
`FileName`	Optional. Sets the name (and optionally, the path) of the log file. This attribute supports UNC or a relative file path (relative to the directory where the Business Importer is executing). `FileName` can be dynamically created using keywords. Valid keywords are: • `[DATE]` is replaced by the date the import was started (formatted as `yyyymmdd`). • `[TIME]` is replaced by the time the import was started (formatted as `hhmmss`). • `[IMPORT NAME]` is replaced by the value of the `Name` attribute of the `Import` element containing this `Log`. If `Output="File"` but the `FileName` is not specified, the default is to write the log file in the directory where the Business Importer is executing, with the name `MBI.log.txt`.
`LogLevel`	Sets the logging level to control logging output. Enabling logging at a given level, also enables logging at all higher levels. Valid values (organized from highest to lowest) are: • `Silent` • `Critical` • `Errors` • `Warnings` • `Information` • `Debug`.

More detailed tracing (written to the database) is also available for each import: see Detailed Tracing.

4

FlexNet Data Domain Interface (DDI)

The FlexNet DDI is simply a set of standard and predefined business adapters that can be used to import data into the central compliance database of FlexNet Manager Suite. They can also be used as starting points for developing your own custom business adapters. This set of standard adapters is maintained over time by Flexera for compatibility with future releases of FlexNet Manager Suite.

Each consists of a sample XML file to configure the adapter, along with a matching CSV file that shows the compatible set of properties for each object.

By populating the sample CSV files with data, you can use these adapters as delivered, or you can customize them to meet specific needs. If you are customizing the adapters, keep copies of the originals and document your changes, since new versions of FlexNet Manager Suite may require additional work to migrate your customizations to the latest versions of the adapters released with FlexNet Manager Suite.

The DDI described in the chapter is for use on the central application server of an on-premises implementation of FlexNet Manager Suite. A similar set (allowing for disconnected operation) is installed automatically on each inventory beacon.

Installation and Set-Up

The details for the DDIs are slightly different in disconnected mode, running on an inventory beacon, and (for on-premises implementations only) running on connected mode on your central application server.

Disconnected mode on an inventory beacon

No installation or set up is required. The DDIs are installed (along with the Business Importer and the Business Adapter Studio) as a default part of the installation of the FlexNet Beacon software on the inventory beacon. The files are stored in the following directories:

Directory	Notes
`C:\ProgramData\Flexera Software\Beacon\DDI`	The adapter XML files for your use or customization
`C:\ProgramData\Flexera Software\Beacon\ExampleCSV`	Example CSV files matching the above adapter XML files.

In disconnected mode, you cannot use any SQL statements in your business adapter. The example adapter files in the above folder are for use in disconnected mode on your inventory beacons, and do not support connected mode access to your compliance database.

Connected mode on the application server

Connected mode allows greater flexibility, in that you can include SQL statements in your adapter.

You may wish to install the Business Adapter Studio on your application server, for which see the chapter *Installing a Free-Standing Studio* in the *Installing FlexNet Manager Suite 2019 R2 On-Premises* PDF file.

The DDIs are installed by default on your application server. The default location of the DDI files is `C:\ProgramData\Flexera Software\Beacon\Master\DDI`. (If you have a larger implementation with separate servers, this path is on your batch server.)

The DDIs rely on CSV (Comma Separated Values) example files that are installed by default in `C:\ProgramData\Flexera Software\Beacon\ExampleCSV` on the same server. It is good practice to keep these templates unedited, as master copies.

Each DDI addresses a specific object in the central compliance database for FlexNet Manager Suite, and each uses a separate CSV file:

Database object	DDI Name	CSV File
Asset	Asset	`Asset.csv`
Business Unit	BusinessUnit	`BusinessUnit.csv`
Category	Category	`Category.csv`
Contract	Contract	`Contract.csv`
Cost Center	CostCenter	`CostCenter.csv`
License	License	`License.csv`
Location	Location	`Location.csv`
Purchase Order	PurchaseOrder	`PurchaseOrder.csv`
Software License Key	SoftwareLicenseKey	`SoftwareLicenseKey.csv`
Term and Condition (for contracts)	TermAndCondition	`TermAndCondition.csv`
User	User	`User.csv`
Vendor	Vendor	`Vendor.csv`

The Business Importer is also installed by default (its path is shown below as *`InstBIDir`*). Once in use, the Business Importer creates additional child directories as described below. File and directory names are not case sensitive:

Directory	Notes
`InstBIDir`	Location of the Business Importer executable (`MGSBI.exe`). The default value is `C:\Program Files (x86)\Flexera Software\Inventory Beacon\DotNet\bin\`. ***Tip:*** *The following recommended folders should be created manually for by the Business Importer when run in connected mode on the central application server (or batch server).*
`InstBIDir``\Log`	Location of the log files generated by imports using custom adapters and CSV files.
`InstBIDir``\Data`	Location of the CSV files that are ready to import, after they have been populated with data. As you populate a CSV file with data ready to import, save it into this *`InstalledBIDir`*`\Data` directory.
`InstBIDir``\Templates\DDI`	Location of the template DDI files (adaptor XML files). ***Tip:*** *You may create this folder and store unmodified XML adapter files here, if you wish. It is good practice to keep these templates unedited, as master copies. Edited and operational XML files move to the* *`InstBIDir`* *directory, alongside the Business Importer executable (*`MGSBI.exe`*).*
`InstBIDir``\Templates\ExampleCSV`	Location of the matching CSV templates. ***Tip:*** *You may create this folder and copy the example CSV files here manually, if you wish. It is good practice to keep these templates unedited, as master copies. Edited and populated CSV files move to the* `..\..\`*`Data`* *directory.*

Using the DDIs

The description of the CSV files for each DDI is provided in the following sections. In each listing, the value(s) marked as **Key** are used for matching incoming data against existing records in the compliance database. In this format, the files are for use on the application server of an on-premises implementation.

Each DDI contains a large number of properties, the most commonly used for each database object. Some properties of the corresponding database objects are not represented. The chapter on The FlexNet Business Importer Data Model documents the full data model available to the Business Importer, including the mapping of each property to the objects and properties in the compliance database. You can also find more details about the compliance database tables and properties in the separate *FlexNet Manager Suite Schema Reference* PDF file.

Custom properties have not been added to the DDIs.

Each input CSV file must contain, at a minimum, the key properties used for matching objects in the compliance database, and the properties listed as mandatory. All the other properties are optional.

The properties can be set in any order within the input file.

If currencies are used, all amounts in a single CSV import file must be in the same currency, applying the same exchange rate snapshot against the default currency configured in FlexNet Manager Suite.

For handling of dates, see Entering Dates in the CSV Templates.

Keep in mind that the web interface for FlexNet Manager Suite focuses on individual purchases; but the database stores these in purchase order lines, and has separate records for the purchase orders (headers) themselves. If purchase orders are to be imported, the DDIs as supplied assume the standard configuration where FlexNet Manager Suite is configured to automatically calculate the PO totals.

Any DDI can be customized. However, keep in mind that each release of FlexNet Manager Suite republishes the DDIs, with or without changes to suit the new version. For this reason, your customization process should include saving a separate copy of the original template (so that you can check the latest release against your version to identify any factory-made changes); as well as storing a backup copy of your customized DDI that cannot be over-written during a product upgrade. You may then need to restore your customizations into the new version of the product.

Before running any DDI, check the connection string in the corresponding XML file. By default, the connection string is set to run against database called `FNMP` on a local SQL Server. If the DDI is run from a remote machine, or if the database name is not the default, edit the XML files and adjust the values to the local configuration.

You can run any DDI once from the command line, or you can use Windows Scheduler to automate the data import at regular intervals. For details of the command line interface, see Command Line for Business Importer. In summary, to run any DDI, change into the installation directory where the Business Importer executes, and use a command line like the following:

```
MGSBI.exe /ConfigFile=XMLFileName /Import=ImportName
```

For example, to import a CSV file to update the Location enterprise groups:

```
MGSBI.exe /ConfigFile=location.xml /Import=Location
```

Entering Dates in the CSV Templates

In handling imports of CSV files, FlexNet Business Importer uses the default .NET data parsing algorithm. This uses the current culture setting (Regional Settings) for the account running the Business Importer on the application server doing the import to determine acceptable date/time formats.

This means that you can enter dates in CSV templates using the normal format in which dates are entered in the FlexNet Manager Suite web interface.

If in doubt, use what .NET calls the `UniversalSortableDateTimePattern`:

```
YYYY-MM-DD
YYYY-MM-DD HH:mm:ssZ
```

Examples:

```
2016-03-28
2017-09-21 11:14:00Z
```

```
2018-11-03 23:00:00Z
```

For information only, other formats in U.S. English that are acceptable for .NET and usable for data entry in FlexNet Manager Suite include:

Name	Format / Example
`FullDateTimePattern`	*`dddd, MMMM dd, yyyy h:mm:ss tt`* Example: `Monday, May 28, 2017 11:35:00 AM`
`LongDatePattern`	*`dddd, MMMM dd, yyyy h:mm:ss tt`* Example: `Monday, May 28, 2017 11:35:00 AM`
`RFC1123Pattern`	*`ddd, dd MMM yyyy HH:mm:ss`* `GMT` Example: `Mon, 28 May 2016 11:35:00 GMT`
`ShortDatePattern`	*`M/d/yyyy`* Example: `5/28/2017`
`SortableDateTimePattern`	*`yyyy-MM-ddTHH:mm:ss`* Example: `2017-05-28T18:35:00`

Asset CSV

Asset records correspond to a physical asset register. They are separate from the records of inventory devices, just as an asset register is separate from the physical devices it lists. The asset records and inventory device records are linked. (In the following table, long attribute names have been wrapped for publication, but in the CSV files each is one unbroken string.)

Table 1: Spreadsheet columns for `Asset` DDI

Column	Details
`AcquisitionMode`	*Type:* String. Optional. Maximum: 100 The acquisition mode. Standard modes are: • `Purchased` • `Leased` • `Rented` • `Loaned`. If no value is specified for a new asset, no value is set in the database record. For single tenant on-premises implementations, new acquisition modes will be created for non-existing values. (Not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)
`AssetName`	*Type:* String. Mandatory. Maximum: 256 The name of the computer or a brief description of the asset.

Column	Details
AssetStatus	*Type:* String. Optional. Maximum: 100 The asset's status. Standard values are: • Purchased • In Storage • Installed • Retired • Disposed • Other. If no value is specified for a new asset, the value will set to Purchased. For single tenant on-premises implementations, new asset status values not yet in the compliance database but found in the incoming data will be created automatically. (Not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.) ***Tip:*** *Setting the* AssetStatus *to any of* In Storage, Retired, *or* Disposed *for import through the Business Importer excludes the device linked to the asset record from license consumption calculations. Setting* AssetStatus *by this method does not:* • *Remove installation records for the linked device* • *Remove individual license allocations for the linked device.* *Those effects rely on the status being changed through the web interface for FlexNet Manager Suite.*
AssetTag	*Type:* String. Optional. Maximum: 256 The asset tag that identifies the physical device. This may be something like a barcode number.

Column	Details
AssetType	*Type:* String. Optional. Maximum: 64 The asset type. Standard types are: • Workstation • Server • Monitor • Desk • Chair • Printer • Router • Switch • Telephone • Cell phone • Laptop • Mobile Device. If no value is specified for a new asset, the Workstation value is used. For single tenant on-premises implementations, new asset types are created for incoming non-existing values (so beware of typographical errors): for behavior and presentation, the Workstation template will be applied to these new types. (Not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)
AssignToUser	*Type:* String. Optional. Maximum: 128 The employee number of the asset assigned user. The user must exit in the compliance database; if not, the value will be discarded.
BusinessUnit	*Type:* String. Optional. Maximum: 500 (64) The business unit of the asset: the name of the last level or some of the last levels or all levels. Level and sub-levels must be separated by the forward slash [/] character. The maximum size for each level is 64 characters.
Category	*Type:* String. Optional. Maximum: 500 (64) The category of asset: the name of the last level or some of the last levels or all levels. Level and sub-levels must be separated by the forward slash [/] character. The maximum size for each level is 64 characters.
ChargesAmount	*Type:* Float. Optional. Maximum: - Amount to be charged back for the use of this asset.

Column	Details
`ChargesFrequency`	*Type:* String. Optional. Maximum: 100 The frequency with which the charge back price is charged. Standard values are: • `None` • `Weekly` • `Monthly` • `Quarterly` • `Yearly` • `Lump Sum`. If no value is specified for a new asset, no value is set in the compliance database. For single tenant on-premises implementations, new period descriptions will be created for non-existing values. (Not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)
`Comments`	*Type:* String. Optional. Maximum: 4000 Comments entered about the asset.
`ContractNo`	*Type:* String. Optional. Maximum: 60 The contract number (foreign key) for any contract attached to the asset.
`CostCenter`	*Type:* String. Optional. Maximum: 500 (64) The cost center of the asset: the name of the last level or some of the last levels or all levels. Level and sub-levels must be separated by the forward slash [/] character. The maximum size for each level is 64 characters.
`CurrencyCode`	*Type:* String. Optional. Maximum: 32 The currency code must match an existing currency code in. See the list in Currency Codes.
`CurrencyDate`	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. Date the exchange rate is valid.
`CurrencyRate`	*Type:* Float. Optional. Maximum: - The currency exchange rate against the default currency value set in **System Defaults**.
`CurrencySnapshotName`	*Type:* String. Optional. Maximum: 256 The name of the conversion rate snapshot to be applied to currencies in this import, when converted to the default currency. This must be an exact match for an entry in the `SnapshotName` column of the `CurrencyRateSnapshot` table of the compliance database.

Column	Details
DeliveryDate	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The delivery date.
Depreciation CurrentValue	*Type:* Float. Optional. Maximum: - The current value of the asset.
DepreciationMethod	*Type:* String. Optional. Maximum: 100 The depreciation method. Standard values are: • Straight Line • Residual Value. If no value is specified for a new asset, no value is set in the compliance database. For single tenant on-premises implementations, new depreciation methods will be created for non-existing values. (Not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)
DepreciationPeriod	*Type:* Integer. Optional. Maximum: - The depreciation period (in years), used only for straight line depreciation.
DepreciationRate	*Type:* Float. Optional. Maximum: - The percentage per year that the asset depreciates, expressed as a decimal fraction. For example, 33% p.a. should be represented as `0.33`. This property is used only for residual value depreciation.
Depreciation ResidualValue	*Type:* Float. Optional. Maximum: - The residual value of the asset.
DisposalDate	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The date the asset was disposed of.
EndOfLifeRecipient	*Type:* String. Optional. Maximum: 128 The person or organization who received the asset when it was disposed of.
InstallationDate	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The date the asset was installed.
InventoriedBy Electronic	*Type:* String. Optional. Maximum: 64 The name of the tool used to perform inventory on the assets. This property is overwritten by inventory imports.
InventoriedBy Physical	*Type:* String. Optional. Maximum: 64 The name of the person or tool that performed the last manual inventory.
InventoryDate Electronic	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The date the asset last had inventory reported through software. This property is overwritten by inventory imports.

Column	Details
`InventoryDatePhysical`	*Type:* Date. Optional. The date the asset last had inventory updated (entered) manually.
`LeaseAgreement`	*Type:* String. Optional. Maximum: 200 The contract name of the lease agreement for this asset. This must be an exact match for a `ContractName` property of an entry in the `Contract` table of the compliance database. (For consistent data, this same contract should have a ContractTypeID of 2, which represents a lease agreement.)
`LeaseBuyoutCost`	*Type:* Float. Optional. Maximum: - The cost to purchase this asset at the conclusion of the lease agreement. Depending on the agreement, this may be specified as $1 (a 'dollar buyout'), a price specified in the agreement, or the 'fair market value' of the asset at the expiry of the lease.
`LeaseEndDate`	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The date that the current lease on this asset runs out. (For early termination, you may prefer to use `LeaseTerminationDate`.)
`LeaseNo`	*Type:* String. Optional. Maximum: 60 The contract number of the lease agreement for this asset.
`LeasePeriodicPayment`	*Type:* Float. Optional. Maximum: - The value of recurring payments for this individual asset as defined in its lease agreement. This is sometimes called the 'lease rate'.
`LeasePeriodType`	*Type:* String. Optional. Maximum: 1000 The frequency with which recurring payments are made. The value should match a `PeriodTypeDefaultValue` in the `PeriodType` table in the compliance database. Standard values are: • Weekly • Monthly • Quarterly • Yearly • Lump Sum. For single tenant on-premises implementations, new period types will be created for non-existing values. (Not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)

Column	Details
LeasePrice	*Type:* Float. Optional. Maximum: - You may use this field to record a total liability under a lease agreement, or other purposes as you require. Keep in mind that you have separate fields for: • The regular rental payments for the assets covered by the lease (see LeasePeriodicPayment) • A buyout figure to take over ownership of the assets at the expiry or termination of the lease (see LeaseBuyoutCost)
LeaseStartDate	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The date that the lease on this asset became effective. In general, this is the date from which you may use rented assets, and from which payment schedules are measured.
LeaseTerminationDate	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The date that the lease on this asset was terminated. Normally, use this field for early termination of the lease. If a lease expires in the normal course of business, you may prefer to use LeaseEndDate.
LeaseTerminationReason	*Type:* String. Optional. Maximum: 100 The reason that the lease for this asset was terminated. Value must be matched in the DefaultValue column of the LeaseEndReason table of the compliance database. Standard values are: • Lease Ended - Asset Returned • Early Termination - Asset Returned • Buyout • Early Buyout • Trade. For single tenant on-premises implementations, lease end reasons will be created for non-existing values. (Not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)
Location	*Type:* String. Optional. Maximum: 500 (64) The location of the asset. The value can be specified as a single description level or as levels and sub-levels separated by the forward slash [/] character. The maximum size for each level is 64 characters.
ManufacturerName	*Type:* String. Optional. Maximum: 200 The name of the manufacturer.
ManufacturerPartNo	*Type:* String. Optional. Maximum: 100 The manufacturer's part number.

Column	Details
ModelNo	*Type:* String. Optional. Maximum: 200 The model number.
ResalePrice	*Type:* Float. Optional. Maximum: - The amount the asset was sold for.
RetirementDate	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The date the asset was retired.
RetirementReason	*Type:* String. Optional. Maximum: 100 The reason the asset was disposed of. Standard values are: • Lost • Stolen • Disposed • Sold • Donated • Broken. If no value is specified for a new record of an asset, no value is set in the compliance database. For single tenant on-premises implementations, retirement reasons will be created for non-existing values. (Not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)
SerialNo	*Type:* String. Key. Mandatory. Maximum: 150 The asset's serial number. Used to link to incoming records of inventory devices.
WarrantyEndDate	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The warranty expiration date.
WarrantyType	*Type:* String. Optional. Maximum: 100 The type of warranty. Standard values are: • None • One Year on site • Three years on site. If no value is specified for a new asset, the value is set to None. For single tenant on-premises implementations, new asset warranty types will be created for non-existing values. (Not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)

Column	Details
WrittenOffValue	*Type:* Float. Optional. Maximum: - The value of the asset at time of retirement/disposal.

Business Unit CSV

The Business Unit object is one of four types of enterprise groups in FlexNet Manager Suite: locations, business units, cost centers and categories. Only the name of an enterprise group is exposed in the web interface, but additional properties exist in the database, and can be used in reports and processes.

Table 2: Spreadsheet columns for Business Unit DDI

Column	Details
BusinessUnitName	*Type:* String. Mandatory. Key. Maximum: 64 The name of the business unit. Levels and sub levels must be separated by the forward slash [/] character. The maximum size is 64 for each level, 500 for the full name of an enterprise group (all level names separated by a [/] character.
Description	*Type:* String. Optional. Maximum: 4000 Comments associated with the business unit.

Category CSV

In database terms, the Category object is one of four types of enterprise groups in FlexNet Manager Suite: locations, business units, cost centers and categories.

Table 3: Spreadsheet columns for Category DDI

Column	Details
CategoryName	*Type:* String. Key. Mandatory. Maximum: 64 (500) The name of the category. Levels and sub levels must be separated by the forward slash [/] character. The maximum size is 64 for each level, and 500 for the full path to a category (all level names separated by forward slash [/] characters).
Description	*Type:* String. Optional. Maps to the Comments column of the GroupEx table in the compliance database.

Contract CSV

Contracts may include leases, maintenance agreements, purchase agreements, and the like. Some contracts may control product use rights on software licenses.

Table 4: Spreadsheet columns for `Contract` DDI

Column	Details
`BusinessUnit`	*Type:* String. Optional. Maximum: 500 (64) The business unit of the contract: the name of the last level or some of the last levels or all levels. Level and sub levels must be separated by the forward slash [/] character. The maximum size for each level is 64 characters.
`Category`	*Type:* String. Optional. Maximum: 500 (64) The category given to the contract: the name of the last level or some of the last levels or all levels of categories. Level and sub levels must be separated by the forward slash [/] character. [/] character. The maximum size for each level is 64 characters.
`Comments`	*Type:* String. Optional. Maximum: 4000 Comments recorded about the contract
`ContractDescription`	*Type:* String. Mandatory. Maximum: 100 The contract name, usefully entered to allow easy identification in lists.
`ContractNo`	*Type:* String. Key. Mandatory. Maximum: 60 The contract number or identifying code.
`ContractStatus`	*Type:* String. Optional. Maximum: 100 The status of the contract. Standard status values are: • `Active` • `Archived` • `Draft` • `Suspended` • `Cancelled` • `Expired` • `Completed`. If no value is specified for a new contract, the `Active` value is used. For single tenant on-premises implementations, new status values will be created for non-existing values. (Not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)

Column	Details
`ContractType`	*Type:* String. Optional. Maximum: 100 The type of the contract. Standard types are: • `General` • `Lease` • `Hardware Maintenance and Support` • `Software License` • `Software Maintenance and Support` • `Blanket Purchase` • `Consulting Services` • `Insurance` • `Rent` • `Subscription` • `Microsoft Business and Services Agreement` • `Microsoft Select License Agreement` • `Microsoft Select Plus Agreement` • `Microsoft Select License Enrollment` • `Microsoft Select Plus Affiliate` • `Microsoft Enterprise Agreement` • `Microsoft Enterprise Subscription Agreement.` If no value is specified for a new license, the `General` value will be used. For single tenant on-premises implementations, new contract type will be created for non-existing values. For appearance and behavior, the `General` contract template is applied to these licenses. (Not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)
`CostCenter`	*Type:* String. Optional. Maximum: 500 (64) The cost center of the contract: the name of the last level or some of the last levels or all levels. Level and sub levels must be separated by the forward slash [/] character. [/] character. The maximum size for each level is 64 characters.
`CurrencyCode`	*Type:* String. Optional. Maximum: 32 The currency code must match an existing currency code in ECM. See list in Currency Codes.

Column	Details
CurrencyDate	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. Date the rate is valid.
CurrencyRate	*Type:* Float. Optional. Maximum: - Currency rate against the default currency value set in **System Defaults**.
CurrencySnapshotName	*Type:* String. Optional. Maximum: 256 The name identifying the currency snapshot with conversion rates that apply to currency values in this contract record.
DocumentUNC	*Type:* String. Optional. Maximum: 4000 UNC path to the contract document.
DocumentURL	*Type:* String. Optional. Maximum: 4000 URL to the contract document
Evergreen	*Type:* Boolean. Optional. Default: False. Values may be: • Zero (or an empty cell): false, the default. • One (1): true. When this column evaluates to true, the contract never expires.
ExpiryDate	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The end date of the contract: if no end date is present, the contract will be set to never expire.
GlobalAmount	*Type:* Float. Optional. Maximum: - The overall price of the contract. This may be an upfront payment, or a total value.
LastRenewedDate	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The date on which the contract was last renewed.
Location	*Type:* String. Optional. Maximum: 500 (64) The location of the contract: the name of the last level or some of the last levels or all levels. Level and sub levels must be separated by the forward slash [/] character. The maximum size for each level is 64 characters.
ManagerFullName	*Type:* String. Optional. Maximum: 64 The full name of the manager responsible for the contract. Used to populate the ComplianceResponsibility column of the central database.
MasterContractNo	*Type:* String. Optional. Maximum: 60 The master contract number that is the master of this contract.
MonthlyAmount	*Type:* Float. Optional. Maximum: - The price of any monthly payment associated with the contract.

Column	Details
`NextRenewalDate`	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The date at which the contract is due to be renewed.
`OwnerFullName`	*Type:* String. Optional. Maximum: 64 The owner's full name of the contract.
`ReviewDate`	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The date at which a contract should be reviewed prior to its expiry date. May mark the start of a period when renegotation or renewal is possible.
`SignatoryFullName`	*Type:* String. Optional. Maximum: 64 The signatory's full name of the contract.
`StartDate`	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The start date of the contract.
`VendorName`	*Type:* String. Optional. Maximum: 64 The name of the vendor witch the contract agreement has been made.

Cost Center CSV

The `Cost Center` object is one of four types of enterprise groups in FlexNet Manager Suite: locations, business units, cost centers and categories. Only the name of an enterprise group is exposed in the web interface, but additional properties exist in the database, and can be used in reports and processes.

Table 5: Spreadsheet columns for `Cost Center` DDI

Column	Details
`CostCenterName`	*Type:* String. Key. Maximum: 64 The name of the enterprise group. Levels and sub levels must be separated by the forward slash [/] character. The maximum size is 64 characters for each level, and 500 for the full name of an enterprise group (all level names separated by forward slash [/] characters).
`Description`	*Type:* String. Optional. Maximum: 4000 Comments associated with the cost center.

License CSV

The `License` object is central to software asset management. This is the database object where consumption of entitlements is recorded.

Table 6: Spreadsheet columns for `License` DDI

Column	Details
`BusinessUnit`	*Type:* String. Optional. Maximum: 500 (64) The business unit of the asset: the name of the last level or some of the last levels or all levels. Level and sub-levels must be separated by the forward slash [/] character. The maximum size for each level is 64 characters.
`Category`	*Type:* String. Optional. Maximum: 500 (64) The category of asset: the name of the last level or some of the last levels or all levels. Level and sub-levels must be separated by the forward slash [/] character. The maximum size for each level is 64 characters.
`ChargesAmount`	*Type:* Float. Optional. Maximum: - For interdepartmental charge-backs, this is the amount to be charged for each computer on which applications linked to the license is installed.
`ChargesFrequency`	*Type:* String. Optional. Maximum: 1000 The frequency with which charge-back amounts are accounted for. The value should match a `PeriodTypeDefaultValue` in the `PeriodType` table in the compliance database. Standard values are: • Weekly • Monthly • Quarterly • Yearly • Lump Sum. For single tenant on-premises implementations, new period types will be created for non-existing values. (Not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)
`Comments`	*Type:* String. Optional. Maximum: - Comments about the license recorded by an operator. This field may also be used for storing license keys.

Column	Details
`ComplianceStatus`	*Type:* String. Optional. Maximum: 100 The compliance position of this license. The value should match a `StatusDefaultValue` in the `SoftwareLicenseComplianceStatus` table in the compliance database. Standard values are: • Compliant • At Risk • Unknown • Not Tracked. For single tenant on-premises implementations, new compliance status values will be created for non-existing values. (Not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)
`ContractNo`	*Type:* String. Optional. Maximum: 60 The contract number (foreign key) for any contract attached to this license.
`CostCenter`	*Type:* String. Optional. Maximum: 500 (64) The cost center of the asset: the name of the last level or some of the last levels or all levels. Level and sub-levels must be separated by the forward slash [/] character. The maximum size for each level is 64 characters.
`CurrencyCode`	*Type:* String. Optional. Maximum: 32 The currency code must match an existing currency code in. See the list in Currency Codes.
`CurrencyDate`	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. Date the exchange rate is valid.
`CurrencyRate`	*Type:* Float. Optional. Maximum: - The currency exchange rate against the default currency value set in **System Defaults**.
`CurrencySnapshotName`	*Type:* String. Optional. Maximum: 256 The name of the conversion rate snapshot to be applied to currencies in this import, when converted to the default currency. This must be an exact match for an entry in the `SnapshotName` column of the `CurrencyRateSnapshot` table of the compliance database.
`DocumentUNC`	*Type:* String. Optional. Maximum: 4000 UNC path to the license contract document. At your discretion, you may use this to link to a copy of the license agreement, or to a contract (such as a purchasing agreement or a support contract) that is relevant to this license.

Column	Details
DocumentURL	*Type:* String. Optional. Maximum: 4000 URL to the license contract document. At your discretion, you may use this to link to a copy of the license agreement, or to a contract (such as a purchasing agreement or a support contract) that is relevant to this license. This field accepts a complete URL, including the protocol.
Duration	*Type:* String. Optional. Maximum: 100 How long entitlements conveyed by this license persist. The value should match a `DurationDefaultValue` in the `SoftwareLicenseDuration` table in the compliance database. Standard values are: • Perpetual • TimeLimited • Subscription. For single tenant on-premises implementations, new duration values will be created for non-existing values. (Not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)
Edition	*Type:* String. Optional. Maximum: 60 The edition of the license. Typically kept synchronized with the edition of the application whose use is authorized by the license.
ExpiryDate	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The date that rights conveyed by this license expire. Use this for a time-limited license, and perhaps for a subscription license if you so choose (remembering that you will need to maintain the date if the subscription is renewed). Leave this column empty for perpetual licenses.
ExtraEntitlements	*Type:* Integer. Optional. Maximum: - The number of entitlements conveyed by this license, apart from the ones recorded in purchases linked to the license. This is the figure displayed in the **Extra entitlements** field on the **Compliance** tab of the license properties.
LicenseKey	*Type:* String. Optional. Maximum: 256 The multiple-use license key of the license. Only used when the license key type is a multi-use key (for example, an Enterprise key used to cover multiple installs).
LicenseName	*Type:* String. Key. Mandatory. Maximum: 256 Name of the license.

Column	Details
`LicenseStatus`	*Type:* String. Optional. Maximum: 100 The current status of this license in the business lifecycle. The value should match a `DefaultValue` in the `LicenseStatus` table in the compliance database. Standard values are: • Active • Retired • In Stock • Purchased • Received. For single tenant on-premises implementations, new license status values will be created for non-existing values. (Not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)

Column	Details
LicenseType	*Type:* String. Optional. Maximum: 100 The kind of license. The value should match a TypeDefaultValue in the SoftwareLicenseType table in the compliance database. Standard values are: • Enterprise • Device • Node-Locked • User • Concurrent User • Appliance • Client Server • OEM • Evaluation • Run-Time • Device (Processor-Limited) • Site • Named User • Device (Core-Limited) • Core Points • Oracle Processor • Oracle Named User Plus • Processor Points • Oracle Legacy • Enterprise Agreement • SAP Named User • Microsoft Server Processor • CAL • Tiered Device • IBM Processor Value Unit • IBM Authorized User

Column	Details
	• IBM Concurrent User • IBM Floating User • Custom Metric • Processor • IBM Resource Value Unit • IBM User Value Unit • Microsoft Server Core • Oracle User • SAP Package • Microsoft SCCM Client Device • Microsoft SCCM Client User • Microsoft Developer Network. For single tenant on-premises implementations, new license type values will be created for non-existing values. (Not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)
`Location`	*Type:* String. Optional. Maximum: 500 (64) The location responsible for this license. The value can be specified as a single description level or as levels and sub-levels separated by the forward slash [/] character. The maximum size for each level is 64 characters.
`OverrideUnitPrice`	*Type:* Float. Optional. Maximum: - The most recent cost-per-entitlement for this license. This figure is used to calculate exposure or savings when there are no purchases linked to the license. It is displayed in the **Override unit price** field on the **Purchases** tab of the license properties.
`PublisherName`	*Type:* String. Optional. Maximum: 64 The publisher who manufactures and distributes the application attached to this license. The value should match a `VendorName` value in the `Vendor` table in the compliance database. Note that the vendor record must exist in the compliance database before you import this license object.
`ResalePrice`	*Type:* Float. Optional. Maximum: - When the license (and the associated software) has been disposed of by reselling to someone else, this field stores the price for which the software assets were sold.

Column	Details
ResaleRecipient	*Type:* String. Optional. Maximum: 128 When the license (and the associated software) has been disposed of by reselling to someone else, this field identifies the person or organization who took over the software assets.
RetirementDate	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The date that this license was removed from use in your enterprise. Leave this column empty for licenses that are still in use ("active").
RetirementReason	*Type:* String. Optional. Maximum: 100 The reason why this license was removed from active use in your enterprise. The value should match a DefaultValue in the EndOfLifeReason table in the compliance database. Standard values are: • Nothing (leave this column blank) • Lost • Stolen • Disposed • Sold • Donated • Broken. For single tenant on-premises implementations, new retirement reason values will be created for non-existing values. (Not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)
SubjectToTrueUp	*Type:* Boolean. Optional. Default: False. Set this field to True if the license is a true-up license (and so will never be displayed as at risk).
Version	*Type:* String. Optional. Key. Maximum: 60 The version of the license. While strictly speaking the version can be left blank, it forms part of a multi-part key to identify existing license records in the database for possible updates. It is therefore recommended that you provide a version in your CSV import.

Location CSV

The Location object is one of four types of enterprise groups in FlexNet Manager Suite: locations, business units, cost centers and categories. Only the name of an enterprise group is exposed in the web interface, but additional properties exist in the database, and can be used in reports and processes.

Table 7: Spreadsheet columns for `Location` DDI

Column	Details
`LocationName`	*Type:* String. Key. Mandatory. Maximum: 64 (500) The name of the enterprise group. Levels and sub-levels must be separated by the forward slash [/] character. The maximum size is 64 characters for each level, and 500 for the full name of an enterprise group (all level names separated by forward slash [/] characters).
`Description`	*Type:* String. Optional. Maximum: 4000 Comments associated with the location.

Purchase Order CSV

In the web interface for FlexNet Manager Suite, attention is focused on the individual purchase. If multiple purchases were included within a single purchase order, these are related simply by having the same purchase order number. Similarly, this import mainly focuses on the properties of an individual purchase, with just a few of the properties relating to the enclosing purchase order. Purchase order headers, and purchases (purchase order lines), are stored separately in the compliance database: the import process applies the different data items in this DDI to the appropriate tables.

Table 8: Spreadsheet columns for `Purchase Order` DDI

Column	Details
`BusinessUnit`	*Type:* String. Optional. Maximum: 500 (64) The business unit of the purchase order line: the name of the last level or some of the last levels or all levels. Levels and sub-levels must be separated by the forward slash [/] character. The maximum size for each level is 64 characters.
`Category`	*Type:* String. Optional. Maximum: 500 (64) The category of the purchase order line: the name of the last level or some of the last levels or all levels. Levels and sub-levels must be separated by the forward slash [/] character. The maximum size for each level is 64 characters.
`Comments`	*Type:* String. Optional. Maximum: 4000 Comments entered about the purchase. Intended for general comments: for the details of what was purchased, see `Description`.
`ContractNo`	*Type:* String. Optional. Maximum: 64 The contract number attached to the purchase. This may be a purchasing agreement, a maintenance agreement, and so on.

Column	Details
CostCenter	*Type:* String. Optional. Maximum: 500 (64) The cost center of the purchase order line: the name of the last level or some of the last levels or all levels. Levels and sub-levels must be separated by the forward slash [/] character. The maximum size for each level is 64 characters.
CurrencyCode	*Type:* String. Optional. Maximum: 32 The currency code must match an existing currency code in FlexNet Manager Suite. See the list in Currency Codes.
CurrencyDate	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. Date the exchange rate is valid.
CurrencyRate	*Type:* Float. Optional. Currency exchange rate against the default currency set in **System Defaults** in FlexNet Manager Suite.
CurrencySnapshotName	*Type:* String. Optional. Maximum: 256 The name of the conversion rate snapshot to be applied to currencies in this import, when converted to the default currency. This must be an exact match for an entry in the SnapshotName column of the CurrencyRateSnapshot table of the compliance database.
Description	*Type:* String. Optional. Maximum: 250 What was purchased. Keep this entry precise. General comments can be entered separately under Comments.
DocumentUNC	*Type:* String. Optional. Maximum: 4000 UNC to the purchase order (header) document.
DocumentURL	*Type:* String. Optional. Maximum: 4000 URL to the purchase order (header) document.
InvoiceDate	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The date the item was invoiced.
InvoiceNo	*Type:* String. Optional. Maximum: 50 The invoice number for the purchase order.
Location	*Type:* String. Optional. Maximum: 500 (64) The location owning the purchase: the name of the last level or some of the last levels or all levels. Levels and sub-levels must be separated by the forward slash [/] character. The maximum size for each level is 64 characters.

Column	Details
`PartNo`	*Type:* String. Optional. Maximum: 100 The part number of the item purchased. For software items, record the SKU number in this column. Providing the manufacturer's SKU (Stock Keeping Unit) number allows for much greater automation, such as automatically applying product use rights to the associated license.
`POLineNo`	*Type:* Integer. Key. Mandatory. Maximum: - The purchase order line sequence number. This identifies the individual purchase within the purchase order.
`PublisherName`	*Type:* String. Optional. Maximum: 64 The name of the company producing the software.
`PurchaseDate`	*Type:* Date. Mandatory. For formats, see Entering Dates in the CSV Templates. The purchase order date.
`PurchaseOrderNo`	*Type:* String. Key. Mandatory. Maximum: 50 The purchase order number.
`PurchaseQuantity`	*Type:* Integer. Optional. Maximum: - The quantity of identical items bought together in this purchase (the single purchase order line).

Column	Details
`PurchaseType`	*Type:* String. Optional. Maximum: 100 The type of purchase: Standard types are: • `Not Set` • `Software` • `Software Upgrade` • `Hardware` • `Services` • `Other`. If no value is specified for a new purchase order line, the `Not Set` value will be used. ***Tip:*** *In some circumstances, automation treats the `Not Set` value the same way it treats the `Software` value.* For single tenant on-premises implementations, new types will be created for non-existing values. However, it is not recommended to create new values, because specific processing actions are tied to each of the existing types. New types added through the Business Importer have no associated processing. (New types are not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)
`QuantityPerUnit`	*Type:* Integer. Optional. Maximum: - The multiplier applied to each unit of `PurchaseQuantity`. Typically this is `1`, so that the total purchased is identical to the `PurchaseQuantity`; but at other times it may not be. For example, if you purchases 3 license packs (`PurchaseQuantity` is 3), each of which permitted 5 installations (`QuantityPerUnit` is 5), the total number of license entitlements from this purchase is 15.
`RequestDate`	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The date the item was requested.
`RequestNo`	*Type:* String. Optional. Maximum: 60 The request number for the purchase.
`SalesTax`	*Type:* Float. Optional. The total sales tax amount for the purchase. Must be entered in the same currency, matching the `CurrencyCode` setting.

Column	Details
ShippingAndHandling	*Type:* Float. Optional. Maximum: - The total shipping and handling cost for the purchase. Must be entered in the same currency, matching the CurrencyCode setting.
ShippingDate	*Type:* Date. Optional. For formats, see Entering Dates in the CSV Templates. The date the item was shipped.
Status	*Type:* String. Optional. Maximum: 100 The current stage of this purchase in the business cycle. Acceptable values are: • New • Pending • Completed • Cancelled.
UnitPrice	*Type:* Float. Optional. The unit price of the item. Must be entered in the currency matching the CurrencyCode setting.
VendorName	*Type:* String. Optional. Maximum: 64 The name of the reseller against whom the purchase order was raised.

Software License Key CSV

Some licenses require that you track a key or registration number. Key fields are used for matching the key to the license.

Table 9: Spreadsheet columns for Software License Key DDI

Column	Details
LicenseName	*Type:* String. Key. Mandatory. Maximum: 256 The name of the license.
Version	*Type:* String. Key. Mandatory. Maximum: 60 The version of the license.
Edition	*Type:* String. Key. Optional. Maximum: 60 The edition of the license.
KeyValue	*Type:* String. Mandatory. Maximum: 400 The value of the license key.

Term and Condition CSV

Terms and conditions are recorded separately in FlexNet Manager Suite, and linked to the contract(s) to which they apply.

Table 10: Spreadsheet columns for `Term and Condition` DDI

Column	Details
`BeginDate`	*Type:* Date . Mandatory. For formats, see Entering Dates in the CSV Templates. The start date for the term or condition.
`Comments`	*Type:* String. Optional. Maximum: 4000 Comments recorded about the term and condition.
`ContractNo`	*Type:* String. Key. Mandatory. Maximum: 60 The contract number.
`Description`	*Type:* String. Key. Mandatory. Maximum: 256 The description of the term and condition.
`DocumentReference`	*Type:* String. Optional. Maximum: 100 A reference to a document or a section of a document for the term or condition.
`EndDate`	*Type:* Date . Mandatory. For formats, see Entering Dates in the CSV Templates. The end date for the term or condition.
`TermAndConditionType`	*Type:* String. Mandatory. Maximum: 256 The type of term and condition. Standard types are: • `Acceptance Period` • `Price Change` • `Cancellation` • `Renewal` • `Expiry` • `Review` • `Limitation` If no value is specified for a new type of term and condition, the `Limitation` value is used. For a single-tenant on-premises implementation, custom types of term and condition are created for non-existing values, so be careful about typographical errors. (Not supported for multi-tenant implementations for managed service providers.)

User CSV

A user is a person linked to a computer who uses the software installed on it. Do not confuse a user with an operator, who is a person registered to use FlexNet Manager Suite interactively.

Table 11: Spreadsheet columns for `User` DDI

Column	Details
`AccountName`	*Type:* String. Mandatory. Maximum: 64 The primary SAM account name for the user.
`Address_City`	*Type:* String. Optional. Maximum: 200 The city of the user.
`Address_Country`	*Type:* String. Optional. Maximum: 100 The country of the user.
`Address_State`	*Type:* String. Optional. Maximum: 200 The state of the user.
`Address_Street`	*Type:* String. Optional. Maximum: 200 The street of the user.
`Address_Zip`	*Type:* String. Optional. Maximum: 20 The zip of the user.
`BusinessUnit`	*Type:* String. Optional. Maximum: 500 (64) The business unit of the user: the name of the last level or some of the last levels or all levels. Level and sub levels must be separated by the forward slash [/] character. The maximum size for each level is 64 characters.
`CostCenter`	*Type:* String. Optional. Maximum: 500 (64) The cost center of the user: the name of the last level or some of the last levels or all levels. Level and sub levels must be separated by the forward slash [/] character. The maximum size for each level is 64 characters.
`DomainName`	*Type:* String. Mandatory. Maximum: 32 The primary domain name (flat name).
`Email`	*Type:* String. Optional. Maximum: 200 The email address of the user
`EmployeeNo`	*Type:* String. Key. Mandatory. Maximum: 128 The employee number of the user.

Column	Details
EmploymentStatus	*Type:* String. Optional. Maximum: 256 The employment relationship between this user and the enterprise. Values must match the DefaultValue column of the EmploymentStatus table in the compliance database. Standard values include: • Employee • Consultant • Temporary • Part time • Casual. If no value is specified for a new user, the Employee value is used. For a single tenant on-premises implementation, custom employment status values are created for non-existing values imported from the data source. (Custom values are not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)
FaxNo	*Type:* String. Optional. Maximum: 30 The fax number of the user.
FirstName	*Type:* String. Optional. Maximum: 128 The first name of the user. If not provided, this value will be extracted from the FullName property.
FullName	*Type:* String. Optional. Maximum: 64 The full name of the user. If provided, this value will be split during the import into the FirstName, LastName and MiddleName fields. If not provided, it will be built from the values found in FirstName, LastName and MiddleName properties.
InventorySource	*Type:* String. Optional. Maximum: 64 If this user is reported in inventory, the name of the person or tool that performed the last inventory.
JobTitle	*Type:* String. Optional. Maximum: 128 The job title of the user.
LastName	*Type:* String. Optional. Maximum: 128 The last name of the user. If not provided, this value will be extracted from the FullName property.

Column	Details
`Location`	*Type:* String. Optional. Maximum: 500 (64) The location of the user: the name of the last level or some of the last levels or all levels. Level and sub levels must be separated by the forward slash [/] character. The maximum size for each level is 64 characters.
`ManagerEmployeeNo`	*Type:* String. Optional. Maximum: 128 The manager employee number. The manager must already exist in the database; if not, the value is discarded.
`MiddleName`	*Type:* String. Optional. Maximum: 128 The middle name of the user. If not provided, this value will be extracted from the `FullName` property.
`MobileNo`	*Type:* String. Optional. Maximum: 30 The mobile phone number of the user.
`PhoneNo`	*Type:* String. Optional. Maximum: 30 The business phone number of the user.
`UserStatus`	*Type:* String. Optional. Maximum: 256 The status of the user. Standard values include: • `Active` • `Inactive` • `Retired` • `On Leave` • `Pending`. If no value is specified for a new user, the `Active` value is used. For a single tenant on-premises implementation, custom user status values are created for non-existing values imported from the data source. Users with these custom status values are assumed to be “active” (able to consume licenses) for the purposes of compliance calculations. (Custom values are not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)

Column	Details
UserSuffix	*Type:* String. Optional. Maximum: 100 The suffix to the name of the user. Standard suffixes include: • Jr. • Sr. • I • II • III For a single tenant on-premises implementation, custom user suffixes are created for non-existing values imported from the data source. (Custom values are not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)
UserTitle	*Type:* String. Optional. Maximum: 100 The title of the user. Standard titles include: • Mr. • Miss • Mrs. • Ms. • Dr. • Prof. For a single tenant on-premises implementation, custom user titles are created for non-existing values imported from the data source. (Custom values are not supported for multi-tenant implementations for managed service providers, nor for cloud implementations.)

Vendor CSV

The Vendor table is used to store both the publishers of software and the resellers. You may use the same sample CSV file in both cases. Limited information about vendors is available in the web interface for FlexNet Manager Suite, but additional columns are available in the database, and are available for reports and other purposes.

Table 12: Spreadsheet columns for Vendor DDI

Column	Details
Address_City	*Type:* String. Optional. Maximum: 200 The city of the vendor.

Column	Details
`Address_Country`	*Type:* String. Optional. Maximum: 100 The country of the vendor.
`Address_State`	*Type:* String. Optional. Maximum: 200 The state of the vendor.
`Address_Street`	*Type:* String. Optional. Maximum: 200 The street of the vendor.
`Address_ZIP`	*Type:* String. Optional. Maximum: 20 The zip of the vendor.
`Email`	*Type:* String. Optional. Maximum: 200 The email address of the vendor.
`FaxNo`	*Type:* String. Optional. Maximum: 30 The fax number of the vendor.
`ParentName`	*Type:* String. Optional. Maximum: 64 The name of the vendor's parent. The parent vendor must already exist in the database; if not, the value is discarded.
`PhoneNo`	*Type:* String. Optional. Maximum: 30 The phone number of the vendor.
`VendorName`	*Type:* String. Key. Mandatory. Maximum: 64 The name of the vendor.
`Web_Site`	*Type:* String. Optional. Maximum: 200 The web site of the vendor.

Currency Codes

All monetary amounts within a single import must be entered in the same currency. (They must also be at the same exchange rate between that chosen currency and the base currency, as set within FlexNet Manager Suite.) The chosen currency is identified by one of the following supported currency codes.

Table 13: Supported currency codes

Code	Currency
USD	United States Dollar
ARS	Argentine Peso
AUD	Australian Dollar

Code	Currency
BND	Brunei Dollar
BRL	Brazil Real
CAD	Canadian Dollar
CHF	Swiss Franc
CNY	Chinese Yuan
DKK	Danish Krone
EGP	Egyptian Pound
EUR	Euro
FJD	Fiji Dollar
GBP	British Pound
HKD	Hong Kong Dollar
IDR	Indonesian Rupiah
INR	Indian Rupee
JPY	Japanese Yen
KPW	North Korean Won
KRW	South Korean Won
LKR	Sri Lankan Rupee
MXN	Mexican Peso
MYR	Malaysian Ringgit
NLG	Holland Guilder
NOK	Norwegian Krone
NZD	New Zealand Dollar
PGK	Papua New Guinea Kina
PHP	Philippine Peso
RUB	Russian Rouble
SEK	Swedish Krona
SGD	Singapore Dollar
THB	Thai Baht

Code	Currency
TWD	New Taiwanese Dollar
VEB	Venezuelan Bolivar
VND	Vietnam Dong
ZAR	South African Rand

5

Additional Information

This chapter provides additional information about connection strings and working with a variety of file types.

Validating Connection Strings

There are many ways to build and validate connection strings, but here is one of the most popular.

Other than for reading data from directory services, the Business Importer uses OLE-DB connections to read data from external data sources. You can use the following procedure to build and validate an OLE-DB connection string.

To validate a connection string:

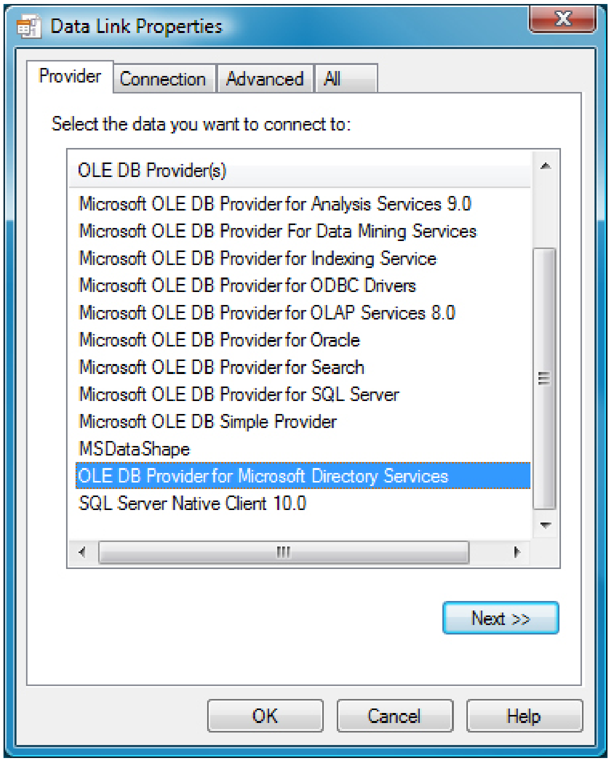
1. In your preferred flat text editor, create a new text file (for example, `test.txt` in the `temp` directory).
2. Make sure Windows Explorer does not hide the extension of the file (**Tools > Folder Options > Views**), and rename the text file by changing the `.txt` extension to `.udl` (for example, `test.udl`).
3. Depending on the architecture of your computer:
    - On a Windows 32-bit machine, double-click on the file.
    - On a 64-bit machine, open a Command Window and use the following command line (wrapped for publishing: enter all on one line):

      ```
      C:\Windows\syswow64\rundll32.exe
           "C:\Program Files (x86)\Common Files\System\Ole DB\oledb32.dll",
           OpenDSLFile FilePathAndUDLFileName
      ```

      For example:

      ```
      C:\Windows\syswow64\rundll32.exe
           "C:\Program Files (x86)\Common Files\System\Ole DB\oledb32.dll",
           OpenDSLFile C:\temp\test.udl
      ```

    The **Data Link Properties** dialog is displayed.
4. Ensure the **Provider** tab is selected, and choose the appropriate OLE-DB provider for this connection.



Data Link Properties

Select the data you want to connect to:

OLE DB Provider(s)

- Microsoft OLE DB Provider for Analysis Services 9.0
- Microsoft OLE DB Provider For Data Mining Services
- Microsoft OLE DB Provider for Indexing Service
- Microsoft OLE DB Provider for ODBC Drivers
- Microsoft OLE DB Provider for OLAP Services 8.0
- Microsoft OLE DB Provider for Oracle
- Microsoft OLE DB Provider for Search
- Microsoft OLE DB Provider for SQL Server
- Microsoft OLE DB Simple Provider
- MSDataShape
- OLE DB Provider for Microsoft Directory Services
- SQL Server Native Client 10.0

***Tip:** There may be several available OLE-DB drivers to connect to an external data source, depending on which providers are installed on your computer. For instance, connection to SQL Server can be accomplished using any of the following drivers:*

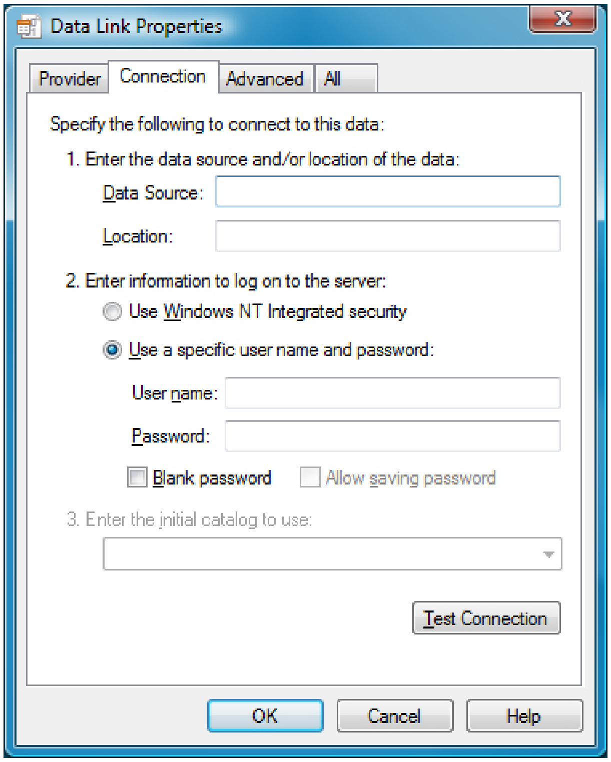
- *Microsoft OLE DB Provider for SQL Server*
- *SQL Server Native Client 10.0*
- *Microsoft OLE DB Provider for ODBC Drivers.*

Notice that in the XML file that configures this connection, the `Type` attribute of the `Import` element must be set to match the provider used. In this instance:

Provider	***Type attribute setting***	***Notes***
Microsoft OLE DB Provider for SQL Server	*`SQLServer`*	*The provider should be omitted from the connection string.*
Microsoft OLE DB Provider for ODBC Drivers	*ODBC*	*The provider should be included in the connection string.*
SQL Server Native Client 10.0	*OLEDB*	*The provider should be included in the connection string.*

For more about connection strings and the `Type` attribute of the `Import` element, see ConnectionString Attribute.

5. Select the **Connection** tab, and enter the details for your connection.



Specify the following to connect to this data:

1. Enter the data source and/or location of the data:

Data Source:

Location:

2. Enter information to log on to the server:

○ Use Windows NT Integrated security

● Use a specific user name and password:

User name:

Password:

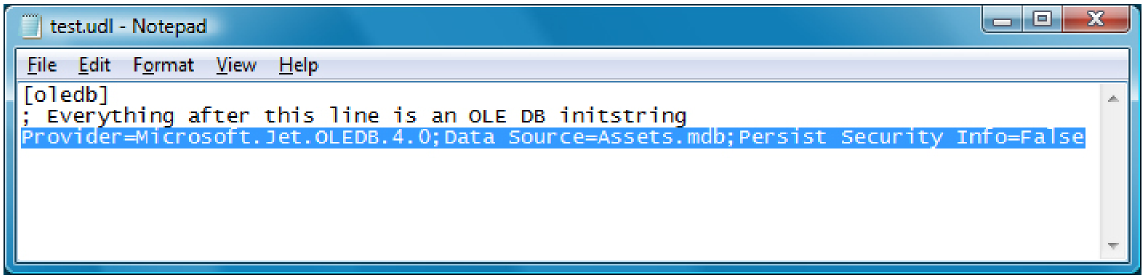
□ Blank password □ Allow saving password

3. Enter the initial catalog to use:

6. When satisfied, test for completeness and accuracy by clicking **Test Connection**.

7. When the connection is validated, click **OK** to save the details into your text file.

8. Open the UDL file in your text editor.



```
[oledb]
; Everything after this line is an OLE DB initstring
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=Assets.mdb;Persist Security Info=False
```

9. For `Oracle` or c/ values of the `Import` element's `Type` attribute, remove the provider value.

10. Copy the [remaining] connection string from the text file, and paste it into the XML adapter file for the Business Importer.

Tip: *This method does not expose all the attributes required for every kind of OLE-DB connector. For instance, extended properties for Excel or CSV files are only available in the* ***Advanced*** *tab of the* ***Data Link Properties*** *dialog. See ConnectionString Attribute for more details on the required values for each connection type.*

Working With Excel Files

Connection strings

For details of connection strings to use with different versions of Excel, see ConnectionString Attribute.

Detecting data types

For best results when importing Excel files, you may need to adjust some registry settings on the computer where the Business Importer is executing. This may be necessary because the OLE-DB driver for Excel automatically assigns the format for each column based on a sampling of the first eight rows of the spreadsheet. This may cause problems in scenarios like the following:

- In a particular column, the first eight rows contain numeric values.
- Therefore the OLE-DB driver sets the column type to `numeric`.
- Further down the column, there are string (text) values in other rows.
- When it encounters these, the driver may either discard the string values, or throw a fatal error.

One possible way to solve the problem is to change the number of rows considered by Excel. Of course, this is only reliable if all rows in the spreadsheet are examined; but on very large spreadsheets, this may have performance impacts.

If you wish to modify the number of rows Excel examines to determine the data type, use `regedit` to locate the following registry key:

- For a 32-bit operating system:

  ```
  HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Jet\4.0\Engines\Excel
  ```

- For a 64-bit operating system:

  ```
  HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Jet\4.0\Engines\Excel
  ```

In this key, the value for `TypeGuessRow` defines the number of rows that Excel reads to determine the format of a column. A value of zero causes Excel to read the full worksheet to evaluate column types.

Working with mixed data types

Depending on the quality of the data and different scenarios that can occur, a spreadsheet column can sometimes contain mixed data types (for example, numeric and string). In this case, data should always be considered as string (with numeric values being numbers that are represented as strings). You can control the behavior of the OLE-DB driver in this case by setting the `ImportMixedType` value (in the same registry key listed above) to `Text`.

If you force this behavior, also ensure that, within the connection string defined in the XML adapter file, the extended properties set the value of `IMEX` to `1`:

```
Provider=Microsoft.ACE.OLEDB.12.0;Data Source=FullPathAndFileName;
Extended Properties=Excel 12.0 Xml;HDR=Yes;IMEX=1
```

One further control

For date/time columns in your source spreadsheet, you can use the `Format` attribute of the `Property` element to clarify the data type. This identifies the format of the data column in the source data, improving conversion for storage in the target database.

The last resort

Even with all these parameters and setting and the additional use of the `Format` attribute of the `Property` element, there are still few cases that cannot be handled by the OLE-DB Excel driver. These cases display as either fatal errors, or as data loss of data corruption. In these instances, the use of the text driver is recommended.

Working With CSV Files

Connection strings

For details of connection strings to use with different versions of Excel, see ConnectionString Attribute.

Detecting data types

Any data surrounded by the text delimiter (double quotation marks [“], is interpreted as text. For example, `"42"` is a text string, and not a number.

The OLE-DB driver for Text automatically assigns the data type of each column based on a sampling of 25 rows. This may generate issues in some cases. For example, if the 25 first rows of particular column include only numeric values, the column is now typed as numeric. If string values exist in the 26th row or beyond, either a fatal error will be generated or values will be discarded.

One possible way to solve the problem is to change the number of rows considered by the driver. Of course, this is only reliable if all rows in the CSV file are examined; but on very large files, this may have performance impacts.

To modify the number of rows Excel examines to determine the data type, use `regedit` to locate the following registry key:

- For a 32-bit operating system:

  ```
  HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Jet\4.0\Engines\Text
  ```

- For a 64-bit operating system:

  ```
  HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Jet\4.0\Engines\Text
  ```

In this key, the value for `MaxScanRows` defines the number of rows that the driver reads to determine the data type of a column. A value of zero causes the driver to read the full worksheet to evaluate column types.

Using Schema.ini

Columns names, data types, character set, and data conversions are specified for the Text OLE-DB driver by using a `schema.ini` file. This file contains the definition of the columns for any text file in the current directory and overwrites all other settings.

Microsoft Windows offers an easy way to generate a default `schema.ini` file based on the existing text files in a directory. For details, see Editing a Schema.ini File.

A CSV example

Assume that an `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 adapter file used to load the assets into the compliance database is as follows:

```
<Import Name="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]">
    <Log Name="NewLog"
        Output="file"
        Loglevel="debug"
        filename="IMPORT_NAME.log.txt"
    </Log>
    <Object Name="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>
```

Editing a Schema.ini File

A `schema.ini` file defines the columns for a text file, overwriting all other settings.

The OLE-DB driver for a text file looks for a `schema.ini` file in the same directory as the text file to be imported. This can define all aspects of the processing of the text file:

- Column names
- Data types for all columns
- The character set
- Data conversions.

The easiest way to work with a `schema.ini` file is to use the Microsoft Windows facilities to create it, and then edit the file to suit your requirements.

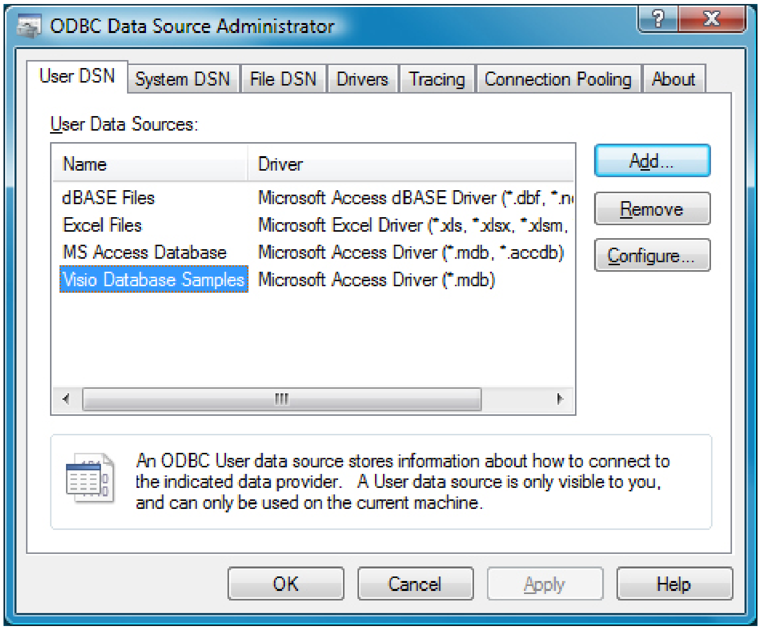
To edit a `schema.ini` file:

1. Depending on your machine architecture:

    - On a 32-bit operating system, open the Control Panel and start ODBC.
    - On a 64-bit operating system, open a Command Window and run:

      ```
      C:\Windows\SysWOW64\Odbcad32.exe
      ```

    The **ODBC Data Source Administrator** dialog appears.



ODBC Data Source Administrator

User DSN | System DSN | File DSN | Drivers | Tracing | Connection Pooling | About

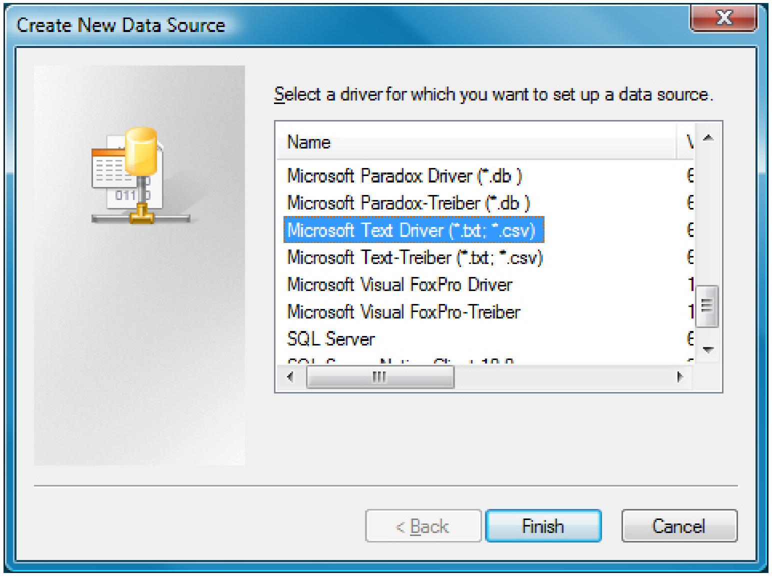
User Data Sources:

Name	Driver
dBASE Files	Microsoft Access dBASE Driver (*.dbf, *.n
Excel Files	Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm,
MS Access Database	Microsoft Access Driver (*.mdb, *.accdb)
Visio Database Samples	Microsoft Access Driver (*.mdb)

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

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

   The **Create New Data Source** dialog appears.



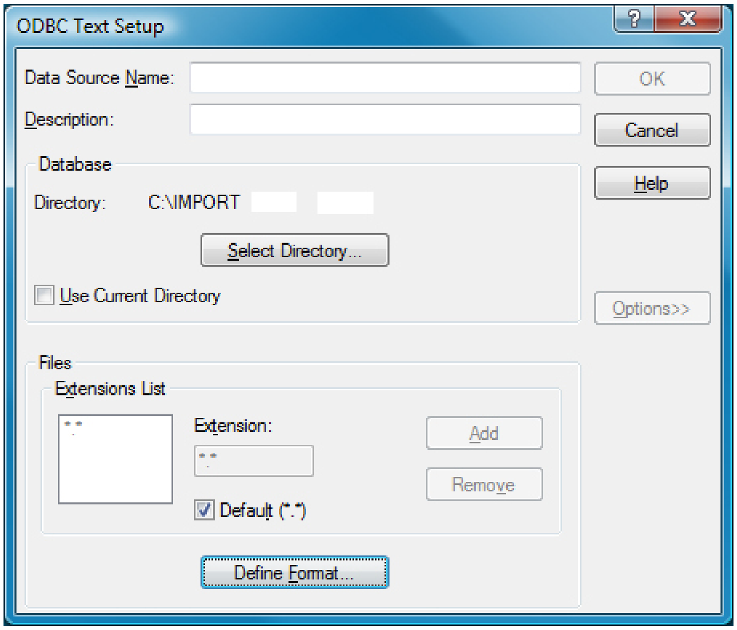
Create New Data Source

Select a driver for which you want to set up a data source.

Name
Microsoft Paradox Driver (*.db)
Microsoft Paradox-Treiber (*.db)
Microsoft Text Driver (*.txt; *.csv)
Microsoft Text-Treiber (*.txt; *.csv)
Microsoft Visual FoxPro Driver
Microsoft Visual FoxPro-Treiber
SQL Server

3. Select `Microsoft Text Driver (*.txt; *.csv)`, and click **Finish**.

   The **ODBC Text Setup** dialog appears.



Data Source Name:

Description:

Directory: C:\IMPORT

□ Use Current Directory

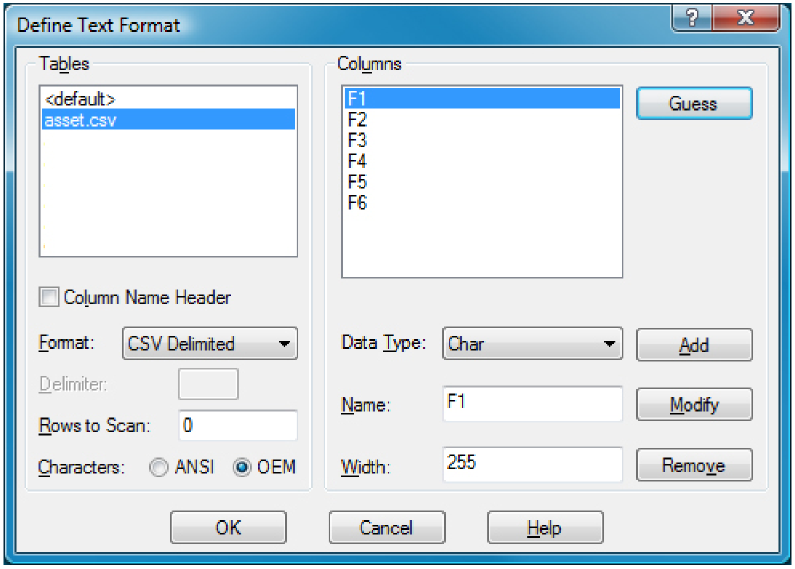
.

Extension: *.*

☑ Default (*.*)

4. Click on the **Select Directory...** button, and select the directory where the text file is stored.
5. Still in the **ODBC Text Setup** dialog, click **Define Format...**.

   The **Define Text Format** dialog appears.



<default>
asset.csv

F1
F2
F3
F4
F5
F6

□ Column Name Header

Format: CSV Delimited

Delimiter:

Rows to Scan: 0

Characters: ○ ANSI ● OEM

Data Type: Char

Name: F1

Width: 255

6. If your text file has a column headers in the first line/row, click **Guess**, and the column names appear in the **Columns** list.

   If required, you can modify column names and formats can be performed in this dialog. Select a column name from the list, and edit the attributes below.
7. When satisfied with your column details, click **OK** in the **Define Text Format** dialog.
8. In the last dialog, click **Cancel** to avoid setting up a new data source.
9. In Windows Explorer, check the directory where the text file resides, and you will find a new `schema.ini` file

there.

10. Edit and modify the schema.ini file using a flat text editor such as Notepad or WordPad.

For more information about configuring a schema.ini file, see http://msdn.microsoft.com/en-us/library/ms709353(VS.85).aspx

Working with Directory Services

The Business Importer can import data from directory services, such as Microsoft Active Directory, or Novell eDirectory. However, configuration and settings are quite unlike other drivers.

Credentials

Usually no login and password are needed as the Business Importer runs under an account that has read access to the directory from which data is to be collected. However, if necessary, login, password, and authentication type can be provided within the ConnectionString attribute using the following format:

```
Login=value;Password=value;AuthenticationType=value
```

The values for AuthenticationType are listed in the following table. Because not all values are supported by all directory services, please check the documentation for your directory service (for example, see https://msdn.microsoft.com/en-us/library/system.directoryservices.authenticationtypes%28v=vs.110%29.aspx).

Authentication Type	Description
Anonymous	No authentication is performed.
Delegation	Enables Active Directory Services Interface (ADSI) to delegate the user's security context.
Encryption	Attaches a cryptographic signature to the message that both identifies the sender and ensures that the message has not been modified in transit.
FastBind	Specifies there will be no attempt to query the Active Directory Domain services objectclass property.
ReadOnlyServer	For a WinNT provider, the Business Importer tries to connect to a domain controller. For Active Directory Domain Services, this flag indicates that a writable server is not required for a serverless binding.
Sealing	Encrypt data using Kerberos.
Secure	Requests secure authentication. This is the default value.
SSL SecureSocketLayer	Attaches a cryptographic signature to the message that both identifies the sender and ensures that the message has not been modified in transit.
ServerBind	Used only if ADsPath includes a server name.
Signing	Verifies data integrity to ensure data received is the same as the data sent.

Authentication Type	Description
None	Set to use basic authentication.

Query structure

Each query to a directory service is made up of two parts:

- A filter, defined in the `Query` attribute of the `Import` element (see Query Attribute).
- A comma-separated list of properties to read, defined in the `PropertiesToLoad` attribute of the `Import` element (see Import Element.)

In defining the filter in the `Query` attribute, each element must be enclosed in parentheses. Expressions can use the relation operators <, <=, =, >=, >. Examples:

```
(objectClass=user)
(lastName>=Davis)
```

Compound expressions are formed with the prefix operators & and | (logical-AND and logical-OR). Examples:

```
(&(objectClass=user)(lastName= Davis))
(&(objectClass=printer)(|(building=42)(building=43)))
```

Additional parameters

The driver supports numerous additional parameters (`server timeout`, `searchscope`, `sizelimit`, and so on). Usually, there is no need to modify the default values set by the connector.

Important: *When using eDirectory, the `pagesize` attribute must be set to `-1`.*

ADSI example

In this example XML adapter file, users and their associated domains and locations are retrieved from Active Directory:

```
<Import
        Name="ADUser"
        Type="ADSI"
        ConnectionString=""
        Query = "(&amp;(objectCategory=user)(objectClass=person)(mail=*))"
        Properties="distinguishedname,sn,cn,givenname,c,company,
                department,division,displayName,employeeID,EmployeeNumber,
                mail,manager,middleName,mobile,title,telephoneNumber,
                SAMAccountName">
        <Log Name="NewLog"
                Output="file"
                Loglevel="warning"
                filename="[DATE][TIME][IMPORT NAME].log.txt">
        </Log>
        <Object Name="Location"
                Type="Location"
                Output="locationid1"
                Update="false"
                Create="True">
                <Property Type="groupCN"
                        Name="Name"
                        Update="No Update"
                        Value="distinguishedname"
                        ValueType="FieldValue"
                        UseForMatching="True"
                        Regex="(?&lt;=OU=).*?(?=,)"
                        RegexOrder ="reverse">
                </Property>
                <Property Type="groupexid"
                        Name="ID"
                        Update="No Update"
                        Value="locationid1"
                        ValueType="FieldValue"
                        UseForMatching="true"
                        MatchingMask="[value]%."
                        MatchingMode ="like"
                        UseNullValueForMatching="removeproperty">
                </Property>
        </Object>

        <Object Name="Compliance Domain"
                Type="compliancedomain"
                Output="compliancedomainoutid"
                Update="False"
                Create="True">
                <Property Name="Qualified Name"
                        Type="QualifiedName"
                        Update="Never"
                        Value="distinguishedname"
                        ValueType="Field Value"
                        UseForMatching="false"
                        Regex="(?&lt;=DC=).*"
                        RegexReplace=",DC="
                        RegexReplaceBy= ".">
                </Property>
                <Property
                        Name="Flat Name"
                        Type="flatname"
                        Update="Never"
                        Value="distinguishedname"
                        ValueType="Field Value"
                        UseForMatching="true"
                        Regex="(?&lt;=DC=.*DC=).*?(?=,DC=)">
                </Property>
        </Object>

        <Object
                Name="User"
                Type="user"
                Output="useroutid"
                Update="true"
                Create="True">
                <Property Name="User Name"
                        Type="username"
                        Update="Always"
                        Value="cn"
                        ValueType="Field Value"
                        UseForMatching="False">
                </Property>
                <Property Name="EmployeeNumber"
                        Type="employeenumber"
                        Update="Always"
                        Value="EmployeeNumber"
                        ValueType="Field Value"
                        UseForMatching="False">
                </Property>
                <Property Name="FirstName"
                        Type="firstname"
                        Update="Always"
                        Value="givenname"
                        ValueType="Field Value"
                        UseForMatching="False">
                </Property>
                <Property Name="MiddleName"
                        Type="middlename"
                        Update="Always"
                        Value="middleName"
                        ValueType="Field Value"
                        UseForMatching="False">
                </Property>
                <Property Name="LastName"
                        Type="lastname"
                        Update="Always"
                        Value="sn"
                        ValueType="Field Value"
                        UseForMatching="False">
                </Property>
                <Property Name="JobTitle"
                        Type="jobtitle"
                        Update="Always"
                        Value="title"
                        ValueType="Field Value"
                        UseForMatching="False">
                </Property>
                <Property Name="BusinessPhoneNumber"
                        Type="BusinessPhoneNumber"
                        Update="Always"
                        Value="telephoneNumber"
                        ValueType="Field Value"
                        UseForMatching="False">
                </Property>
                <Property Name="MobilePhoneNumber"
                        Type="MobilePhoneNumber"
                        Update="Always"
                        Value="mobile"
                        ValueType="Field Value"
                        UseForMatching="False">
                </Property>
                <Property Name="Locationid"
                        Type="locationid"
                        Update="Always"
                        Value="locationid1"
                        ValueType="Field Value"
                        UseForMatching="False">
                </Property>
                <Property Name="Email"
                        Type="Email"
                        Update="Always"
                        Value="mail"
                        ValueType="Field Value"
                        UseForMatching="False">
                </Property>
                <Property Name="SAMAccountName"
                        Type="SAMAccountName"
                        Update="Always"
                        Value="SAMAccountName"
                        ValueType="Field Value"
                        UseForMatching="True">
                </Property>
                <Property Name="ComplianceDomainID"
                        Type="ComplianceDomainID"
                        Update="Always"
                        Value="ComplianceDomainoutID"
                        ValueType="Field Value"
                        UseForMatching="False">
                </Property>
        </Object>
</Import>
```

Working With XML Files

Business Importer can import data from XML files that meet the following requirements:

- The XML file must have a single root element (of any name) that contains all the other elements that will be imported.
- All the child elements of the root node must be identical element types.

Here is a simple example:

```
<AssetsToImport>
    <Asset  AssetLocation="Boston"
            AssetName="WKS-00312"
            AssetDeliveryDate="12/05/2007" />
    <Asset  AssetLocation="Los Angelse"
            AssetName="LTP-02321"
            AssetDeliveryDate="04/08/2009" />
</AssetsToImport>
```

Connection to XML files

To specify the connection in the XML adapter file, the only parameters to set are:

- The data source Type="XML"
- The connection string, which must contain both the path to and name of the source XML file (see example below).

Example XML adapter file for importing from an XML source

In the XML adapter file, the mapping to compliance database properties uses the names of the XML tags in the source file. For example, using the XML source file example shown above, the XML adapter file contains:

```
<Import Name="Asset"
      Type="XML"
      ConnectionString="C:\Assets.xml">
        <Object Name="Asset"
                Type="asset"
                Output="assetoutid"
                Update="true"
                Create="true">
                <Property Name="Short Description"
                        Type="shortdescription"
                        Update="Never"
                        Value="AssetName"
                        ValueType="Field Value"
                        UseForMatching="true"/>
                <Property Name="Location"
                        Type="location"
                        Update="Always"
                        Value="AssetLocation"
                        ValueType="Field Value"
                        UseForMatching="False"/>
                <Property Name="Delivery Date
                        "Type="deliverydate"
                        Update="Always"
                        Value="AssetDeliveryDate"
                        ValueType="Field Value"
                        UseForMatching="False"/>
        </Object>
</Import>
```

Working With Web Services

For the Business Importer to collect data from web services, the following items are required:

- The URL of the web service
- The authentication (if any) required to connect to the web service
- The function of the web service to be called or alternatively the SOAP request
- Whether or not header information is required to call the web service
- The name of the XML element within the response that contains the data.

URL and authentication

With the XML adapter file, the URL and authentication are specified in the `ConnectionString` attribute of the `Import` element. The structure is:

```
ConnectionString="Login=MyLogin;Password=MyPassword;URL=http://MyServer/WebService/
MyWebService.asmx"
```

The function call

The function call within the web service is specified in the `Query` attribute of the `Import` element. It can be specified in two ways:

- You can provide the name of the function
- You can provide the full SOAP message.

If just the name of the function is defined, a standard SOAP request is built and forwarded to the web service. For example, if the following function name is specified:

```
Query="GetAllPurchaseOrders"
```

the following SOAP request is built and transmitted:

```
<?xmlversion="1.0"encoding="utf-8"?>
```

```
<soap:Envelope
        xmlns: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 different syntax or parameters, the full SOAP request can be specified in the `Query` attribute of the `Import` element. The above SOAP request could be represented in the following way. Note the need to escape various characters in the mark-up as XML entities. This value has been wrapped to show similarity with the example above:

```
Query="&lt;?xml version=&quot;1.0&quot;
        encoding=&quot;utf-8&quot;?&gt;
        &lt;soap12:Envelope
                xmlns:xsi=&quot;http://www.w3.org/2001/XMLSchema-instance&quot;
                xmlns:xsd=&quot;http://www.w3.org/2001/XMLSchema&quot;
                xmlns:soap12=&quot;http://www.w3.org/2003/05/soap-envelope&quot;&gt;
                &lt;soap12:Body&gt;
                        &lt;GetAllPurchaseOrders xmlns=&quot;http://tempuri.org/&quot;
/&gt;
                &lt;/soap12:Body&gt;
&lt;/soap12:Envelope&gt;"
```

Supplying header information

If header information is required by the web service, use the `SOAPHeaderValues` attribute of the `Import` element to specify the header information. For example:

```
SOAPHeaderValues="SOAPAction=http://MyServer/WebService/GetAllPurchaseOrders"
```

Decoding the reply

When a response is received from the web service, it includes the full SOAP message made up of multiple XML elements. The Business Importer does not know which of these many elements contains the required data. There are three approaches to solving this dilemma:

- The `SOAPElement` attribute of the `Import` element lets you specify the name of the element containing the required data.
- If `SOAPElement` is not specified, the Business Importer looks for an element with a name concatenating the function name with the string `Result` (in the example above, this produces `GetAllPurchaseOrderResult`).

  Tip: *This relies on the Query attribute being used for just the function name rather than the full SOAP request.*

- If these approaches are unsuccessful, the Business Importer falls back to using the content of the `Body` XML element of the response.

If you would like to specify the `SOAPElement` containing the returned data but do not know its name, the Business Importer with the following command line switches:

```
/testdb=ImportName /log=debug
```

These options write the full SOAP answer to the log, where you can analyze it and identify the load-bearing element, plugging its name back into your XM adapter before the next run.

Customizing the Asset-Device Linkage

Within FlexNet Manager Suite, certain asset types (such as workstations, servers, and other kinds of computer) are normally linked to records of inventory devices (computers). This link brings together software and hardware inventory and the details you record for your asset register.

To support this practice, the normal behavior of the Business Importer when creating new asset records of the appropriate type is to perform a search for existing inventory devices that are available (in this case, 'available' means that the records have not been marked as ignored, and the records are not already attached to another asset). The search compares the asset's serial number against the serial numbers recorded for inventory devices. If the serial number of the new asset record matches the serial number of an available inventory device, the link between them is automatically created. (This check is also performed by FlexNet Manager Suite when new inventory device records are created. This means you can create new asset records as the machines are procured, and once inventory is received from a device with a matching serial number, the link is again created automatically. However, this topic is concerned with behavior when adding an asset record while the inventory device record already exists.)

There may be some circumstances where you want to disable this behavior, or replace it with different behavior. (For example, you may wish to resolve potential conflicts between properties imported with your business adapter, and the same properties that are updated on the asset automatically by FlexNet Manager Suite, using data from a matching inventory device record.) You can achieve this customization using the `CustomComputerMatching` attribute of the `Object` element in the XML adapter file.

`CustomComputerMatching` supports the following values:

Value	Description
`CustomComputerMatching=""`	(Empty). The default behavior described above is executed using the inbuilt logic.
`CustomComputerMatching=" "`	(Spaces). The inbuilt logic is disabled, and no links to existing inventory devices are created during this import.
`CustomComputerMatching="`*`SQL-Value`*`"`	The placeholder is replaced either with the name of a stored procedure or with a SQL statement to be executed against the compliance database during the import.

If you are using the third option, the following literal keywords can be included in your SQL, and are replaced by the values shown during the import:

Keyword	Replaced by
`[TemporaryTableName]`	The name of the temporary or physical table that the Business Importer uses for the import.

Keyword	Replaced by
[OutputField]	The name of the field containing the AssetID values for existing or new asset records created.
[ImportID]	The ID of the record in the ECMImportLog_Import table that is reporting the import.
[ImportObjectID]	The ID of the record n the ECMImportLog_Object table that is reporting on the specific object within the import.

When you are creating specific SQL logic, you can retrieve the list of newly created AssetID values with the following query:

```
Select [OutputField] from [TemporaryTableName] where created =1;
```

It is also possible for the SQL statement or procedure to return the number of inventory devices affected by the import. This number is then logged in the ECMImportLog_Object table.

Here is a sample extract of an XML adapter file calling a stored procedure:

```
<Import Name="Asset"
        Type="CSV"
        ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;
                Data Source=c:\temp;
                Extended Properties='text;HDR=Yes;FMT=CSVDelimited'"
        AccountIsEncrypted="False"
        Query="select * from [asset.csv]">
        <Object Name="Asset"
                Type="asset"
                Output="assetoutid"
                Update="true"
                Create="true"
                CustomComputerMatching="exec MyProcedure [TemporaryTableName],
[OutputField]">
                …
                …
        </Object>
</Import>
```

6

The FlexNet Business Importer Data Model

This chapter documents the data model supported by the Business Importer (and therefore the Business Adapter Studio).

The data model is embodied in an XML file called `FNMPDataModel.ini`, which is automatically synchronized between the central application server and all inventory beacons. The one `.ini` file covers the data model for several different releases of FlexNet Manager Suite. The version documented in this chapter is always the most recent version. To validate the behaviors and settings for earlier supported versions, please compare this chapter with the appropriate model from the `FNMPDataModel.ini` file.

The Asset Object

This item maps to the `Asset` object in the FlexNet Manager Suite compliance database. It supports the addition of custom properties through the Business Importer.

Properties

The `Asset` object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 14: Data model properties for `Asset` object (alphabetical listing)

Property	Details
`Acquisition Mode`	*Type:* `nvarchar`. Maximum: 100. Values of `Acquisition Mode` must match those stored in the `AcquisitionMode` table of the compliance database.
`Asset Name`	*Type:* `nvarchar`. Maximum: 256. Mandatory: adapters must provide values for this column. Maps to `Asset.ShortDescription` in the compliance database.

Property	Details
`Asset Status`	*Type:* `nvarchar`. Maximum: 100. Values of `Asset Status` must match those stored in the `AssetStatus` table of the compliance database.
`Asset Tag`	*Type:* `nvarchar`. Maximum: 256. Maps to `Asset.AssetTag` in the compliance database.
`Asset Type`	*Type:* `nvarchar`. Maximum: 64. Values of `Asset Type` must match those stored in the `AssetType` table of the compliance database.
`Assigned To User ID`	*Type:* `int`. Stores the `AssignToUserID` as a foreign key to the `user` table in the compliance database.
`Category ID`	*Type:* `varchar`. Maximum: 128. Stores the `categoryID` as a foreign key to the `category` table in the compliance database.
`Charges Amount`	*Type:* `float`. Maps to `Asset.ChargeBackPrice` in the compliance database.
`Charges Amount Currency Rate ID`	*Type:* `int`. Stores the `ChargeBackPriceRateID` as a foreign key to the `currencyrate` table in the compliance database.
`Charges Frequency`	*Type:* `nvarchar`. Maximum: 100. Values of `Charges Frequency` must match those stored in the `PeriodType` table of the compliance database.
`Comments`	*Type:* `ntext`. Maps to `Asset.Comments` in the compliance database.
`Corporate Unit ID`	*Type:* `varchar`. Maximum: 128. Stores the `businessunitID` as a foreign key to the `CorporateUnit` table in the compliance database.
`Cost Center ID`	*Type:* `varchar`. Maximum: 128. Stores the `costcenterID` as a foreign key to the `costcenter` table in the compliance database.
`Delivery Date`	*Type:* `datetime`. Maps to `Asset.DeliveryDate` in the compliance database.

Property	Details
`Depreciation Current Value`	*Type:* `float`. Maps to `Asset.DepreciationCurrentValue` in the compliance database.
`Depreciation Current Value Currency Rate ID`	*Type:* `int`. Stores the `DepreciationCurrentValueRateID` as a foreign key to the `currencyrate` table in the compliance database.
`Depreciation Method`	*Type:* `nvarchar`. Maximum: 100. Values of `Depreciation Method` must match those stored in the `DepreciationMethod` table of the compliance database.
`Depreciation Period (Years)`	*Type:* `int`. Maps to `Asset.DepreciationPeriod` in the compliance database.
`Depreciation Rate (0-1)`	*Type:* `decimal`. Maps to `Asset.DepreciationRate` in the compliance database.
`Depreciation Residual Value`	*Type:* `float`. Maps to `Asset.DepreciationResidualValue` in the compliance database.
`Depreciation Residual Value Currency Rate ID`	*Type:* `int`. Stores the `DepreciationResidualValueRateID` as a foreign key to the `currencyrate` table in the compliance database.
`Disposal Date`	*Type:* `datetime`. Maps to `Asset.DisposalDate` in the compliance database.
`End of Life Recipient`	*Type:* `nvarchar`. Maximum: 128. Maps to `Asset.EndOfLifeRecipient` in the compliance database.
`Installed On`	*Type:* `datetime`. Maps to `Asset.InstallationDate` in the compliance database.
`Inventoried By Electronic`	*Type:* `nvarchar`. Maximum: 64. Maps to `Asset.InventoryAgent` in the compliance database.
`Inventoried By Physical`	*Type:* `nvarchar`. Maximum: 64. Maps to `Asset.InventoryAgentManual` in the compliance database.
`Inventory Date Electronic`	*Type:* `smalldatetime`. Maps to `Asset.InventoryDate` in the compliance database.
`Inventory Date Physical`	*Type:* `datetime`. Maps to `Asset.InventoryDateManual` in the compliance database.

Property	Details
Lease Agreement	*Type:* nvarchar. Maximum: 200. Maps to Asset.LeaseName in the compliance database.
Lease Buyout Cost	*Type:* money. Maps to Asset.LeaseBuyoutCost in the compliance database.
Lease Buyout Cost Currency Rate ID	*Type:* int. Stores the LeaseBuyoutCostRateID as a foreign key to the currencyrate table in the compliance database.
Lease End Date	*Type:* datetime. Maps to Asset.LeaseEndDate in the compliance database.
Lease Number	*Type:* nvarchar. Maximum: 120. Maps to Asset.LeaseNo in the compliance database.
Lease Periodic Payment	*Type:* money. Maps to Asset.LeasePeriodicPayment in the compliance database.
Lease Periodic Payment Currency Rate ID	*Type:* int. Stores the LeasePeriodicPaymentRateID as a foreign key to the currencyrate table in the compliance database.
Lease Price	*Type:* money. Maps to Asset.LeasePrice in the compliance database.
Lease Price Currency Rate ID	*Type:* int. Stores the LeasePriceRateID as a foreign key to the currencyrate table in the compliance database.
Lease Start Date	*Type:* datetime. Maps to Asset.LeaseStartDate in the compliance database.
Lease Termination Date	*Type:* datetime. Maps to Asset.LeaseTerminationDate in the compliance database.
Lease Termination Reason	*Type:* nvarchar. Maximum: 100. Values of Lease Termination Reason must match those stored in the LeaseEndReason table of the compliance database.
Lease period type	*Type:* nvarchar. Maximum: 100. Values of Lease period type must match those stored in the PeriodType table of the compliance database.

Property	Details
`Location ID`	*Type:* `varchar`. Maximum: 128. Stores the `locationID` as a foreign key to the `location` table in the compliance database.
`Manufacturer`	*Type:* `nvarchar`. Maximum: 100. Maps to `Asset.Manufacturer` in the compliance database.
`Manufacturer Part No`	*Type:* `nvarchar`. Maximum: 100. Maps to `Asset.ManufacturerPartNo` in the compliance database.
`Model No`	*Type:* `nvarchar`. Maximum: 200. Maps to `Asset.ModelNo` in the compliance database.
`Parent Asset ID`	*Type:* `int`. Stores the `ParentAssetID` as a foreign key to the `asset` table in the compliance database.
`Resale Price`	*Type:* `float`. Maps to `Asset.ResalePrice` in the compliance database.
`Resale Price Currency Rate ID`	*Type:* `int`. Stores the `ResalePriceRateID` as a foreign key to the `currencyrate` table in the compliance database.
`Retirement Date`	*Type:* `datetime`. Maps to `Asset.RetirementDate` in the compliance database.
`Retirement Reason`	*Type:* `varchar`. Maximum: 100. Values of `Retirement Reason` must match those stored in the `EndOfLifeReason` table of the compliance database.
`Serial Number`	*Type:* `nvarchar`. Maximum: 150. Maps to `Asset.SerialNumber` in the compliance database.
`Warranty End Date`	*Type:* `datetime`. Maps to `Asset.WarrantyExpirationDate` in the compliance database.
`Warranty Type`	*Type:* `nvarchar`. Maximum: 100. Values of `Warranty Type` must match those stored in the `AssetWarrantyType` table of the compliance database.
`Written Off Value`	*Type:* `float`. Maps to `Asset.WrittenOffValue` in the compliance database.

Property	Details
Written Off Value Currency Rate ID	*Type:* int. Stores the WrittenOffValueRateID as a foreign key to the currencyrate table in the compliance database.

Data model version 13.2

The Category Object

This item maps to the Category object in the FlexNet Manager Suite compliance database.

Database interaction

If an incoming Category is not matched by an existing record in the compliance database, a new record is created there.

If an incoming Category matches an existing record in the compliance database, details are updated (as allowed for individual properties described below).

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the Category appears in the Enterprise Groups group in the right-click context menu.

Within the Business Adapter Studio, the default name suggested for the Output attribute of this Category object is Category_ID.

Properties

The Category object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 15: Data model properties for Category object (alphabetical listing)

Property	Details
Address - City	*Type:* varchar. Maximum: 200. Maps to Category.address_city in the compliance database.
Address - Country	*Type:* varchar. Maximum: 100. Maps to Category.address_country in the compliance database.
Address - State	*Type:* varchar. Maximum: 200. Maps to Category.address_state in the compliance database.

Property	Details
`Address - Street`	*Type:* `varchar`. Maximum: 200. Maps to `Category.address_street` in the compliance database.
`Address - Zip`	*Type:* `varchar`. Maximum: 20. Maps to `Category.address_zip` in the compliance database.
`Business Phone Number`	*Type:* `varchar`. Maximum: 30. Maps to `Category.businessphonenumber` in the compliance database.
`Description`	*Type:* `ntext`. Maps to `Category.comments` in the compliance database.
`Email Address`	*Type:* `varchar`. Maximum: 200. Maps to `Category.email` in the compliance database.
`Fax Number`	*Type:* `varchar`. Maximum: 30. Maps to `Category.faxphonenumber` in the compliance database.
`ID`	*Type:* `int`. For the `ID` property, the Business Adapter Studio offers a list of the output field names from previous objects in the adapter. Maps to `Category.groupexid` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. In the Business Adapter Studio, SQL-like expressions for pattern matching (contained in the `MatchingMask` attribute of the adapter's XML file) are supported for this property. An exception is the case when the incoming `ID` is null. In these cases, the property is excluded from the matching process, which then relies on other properties of the `Category` object. However, updates to the stored value of `Category.groupexid` are not permitted.
`Name`	*Type:* `varchar`. Maximum: 128. Mandatory: adapters must provide values for this column. Individual groups in the path through the enterprise group hierarchy are separated with the / character. Maps to `Category.groupcn` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. However, updates to the stored value of `Category.groupcn` are not permitted.

Data model version 13.2

The Cloud Instance Object

This item maps to the `CloudServiceInstance` object in the FlexNet Manager Suite compliance database.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the `Cloud Instance` appears in the `Computer` group in the right-click context menu.

Properties

The `Cloud Instance` object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 16: Data model properties for `Cloud Instance` object (alphabetical listing)

Property	Details
`Computer ID`	*Type:* `int`. Mandatory: adapters must provide values for this column. Stores the `computerid` as a foreign key to the `Computer` table in the compliance database.
`Hosted In`	*Type:* `nvarchar`. Maximum: 250. Mandatory: adapters must provide values for this column. Maps to `CloudServiceInstance.ServiceProvider` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there.
`Instance Cloud ID`	*Type:* `nvarchar`. Maximum: 256. Mandatory: adapters must provide values for this column. Maps to `CloudServiceInstance.InstanceCloudID` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there.

Data model version 13.2

The Computer Object

This item maps to the `computer` object in the FlexNet Manager Suite compliance database. It supports the addition of custom properties through the Business Importer.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the `Computer` appears in the `Computer` group in the right-click context menu.

Properties

The `Computer` object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 17: Data model properties for `Computer` object (alphabetical listing)

Property	Details
`Asset ID`	*Type:* `int`. Stores the `AssetID` as a foreign key to the `Asset` table in the compliance database.
`Assigned Chassis Type`	*Type:* `nvarchar`. Maximum: 1000. Values of `Assigned Chassis Type` must match those stored in the `ComputerChassisType` table of the compliance database.
`Assigned User ID`	Stores the `AssignedUserID` as a foreign key to the `User` table in the compliance database.
`Calculated User ID`	Stores the `CalculatedUserID` as a foreign key to the `User` table in the compliance database.
`Category ID`	*Type:* `varchar`. Maximum: 128. Stores the `categoryID` as a foreign key to the `category` table in the compliance database.
`Chassis Number`	*Type:* `nvarchar`. Maximum: 128. Maps to `computer.HostIdentifyingNumber` in the compliance database.
`Chassis Type`	*Type:* `nvarchar`. Maximum: 1000. Values of `Chassis Type` must match those stored in the `ComputerChassisType` table of the compliance database.
`Clock Speed (MHz)`	*Type:* `int`. Maps to `computer.MaxClockSpeed` in the compliance database.

Property	Details
`Compliance Status`	Values of `Compliance Status` must match those stored in the `AssetComplianceStatus` table of the compliance database.
`Computer Name`	*Type:* `nvarchar`. Maximum: 256. Maps to `computer.computername` in the compliance database.
`Computer Type`	*Type:* `nvarchar`. Maximum: 1000. Values of `Computer Type` must match those stored in the `ComplianceComputerType` table of the compliance database.
`Corporate Unit ID`	*Type:* `varchar`. Maximum: 128. Stores the `businessunitID` as a foreign key to the `CorporateUnit` table in the compliance database.
`Cost Center ID`	*Type:* `varchar`. Maximum: 128. Stores the `costcenterID` as a foreign key to the `costcenter` table in the compliance database.
`Disk Space (byte)`	*Type:* `int`. Maps to `computer.TotalDiskSpace` in the compliance database.
`Domain ID`	*Type:* `int`. Stores the `ComplianceDomainID` as a foreign key to the `ComplianceDomain` table in the compliance database.
`Email Address`	*Type:* `nvarchar`. Maximum: 512. Maps to `computer.EmailAddress` in the compliance database.
`Hosted In`	*Type:* `nvarchar`. Maximum: 250. Maps to `computer.ServiceProvider` in the compliance database.
`IMEI`	*Type:* `nvarchar`. Maximum: 512. Maps to `computer.IMEI` in the compliance database.
`IP Address`	*Type:* `nvarchar`. Maximum: 256. Maps to `computer.IPAddress` in the compliance database.
`Inventory Date`	*Type:* `datetime`. Maps to `computer.InventoryDate` in the compliance database.
`Inventory Source`	*Type:* `nvarchar`. Maximum: 64. Maps to `computer.InventoryAgent` in the compliance database.

Property	Details
Inventory Source Type	*Type:* nvarchar. Maximum: 1000. Values of Inventory Source Type must match those stored in the ComplianceComputerInventorySourceType table of the compliance database.
Last Logged On User ID	Stores the ComplianceUserID as a foreign key to the User table in the compliance database.
Location ID	*Type:* varchar. Maximum: 128. Stores the locationID as a foreign key to the location table in the compliance database.
MAC Address	*Type:* nvarchar. Maximum: 256. Maps to computer.MACAddress in the compliance database.
Manufacturer	*Type:* nvarchar. Maximum: 128. Maps to computer.Manufacturer in the compliance database.
Model Number	*Type:* nvarchar. Maximum: 128. Maps to computer.ModelNo in the compliance database.
Number of Cores	*Type:* int. Maps to computer.NumberOfCores in the compliance database.
Number of Display Adapters	*Type:* int. Maps to computer.NumberOfDisplayAdapters in the compliance database.
Number of Hard Drives	*Type:* int. Maps to computer.NumberOfHardDrives in the compliance database.
Number of Network Cards	*Type:* int. Maps to computer.NumberOfNetworkCards in the compliance database.
Number of Processors	*Type:* int. Maps to computer.NumberOfProcessors in the compliance database.
Number of Sockets	*Type:* int. Maps to computer.NumberOfSockets in the compliance database.
Number of Threads	*Type:* int. Maps to computer.NumberOfLogicalProcessors in the compliance database.
Operating System	*Type:* nvarchar. Maximum: 128. Maps to computer.OperatingSystem in the compliance database.

Property	Details
`Partial Number of Processors`	*Type:* `float`. Maps to `computer.PartialNumberOfProcessors` in the compliance database.
`Phone Number`	*Type:* `nvarchar`. Maximum: 256. Maps to `computer.PhoneNo` in the compliance database.
`Processor Type`	*Type:* `nvarchar`. Maximum: 256. Maps to `computer.ProcessorType` in the compliance database.
`RAM (byte)`	*Type:* `int`. Maps to `computer.TotalMemory` in the compliance database.
`Role`	*Type:* `nvarchar`. Maximum: 1000. Values of `Role` must match those stored in the `ComplianceComputerRole` table of the compliance database.
`Serial Number`	*Type:* `nvarchar`. Maximum: 100. Maps to `computer.SerialNo` in the compliance database.
`Service Pack`	*Type:* `nvarchar`. Maximum: 128. Maps to `computer.ServicePack` in the compliance database.
`Status`	*Type:* `nvarchar`. Maximum: 1000. Values of `Status` must match those stored in the `ComplianceComputerStatus` table of the compliance database.

Data model version 13.2

The Computer Connection Object

This item maps to the `compliancecomputerconnection` object in the FlexNet Manager Suite compliance database.

Database interaction

Note: *If an incoming `Computer Connection` is not matched by an existing record in the compliance database, the incoming record is rejected. Creation of new records is not allowed for `Computer Connection` objects.*

Note: *If an incoming `Computer Connection` matches an existing record in the compliance database, the incoming details are ignored. Updating existing records is not allowed for `Computer Connection` objects.*

Behavior in the Business Adapter Studio

***Tip:** The Computer Connection object is not available within the Business Adapter Studio.*

When you create new objects in the Business Adapter Studio, the `Computer Connection` appears in the `Computer` group in the right-click context menu.

Properties

The `Computer Connection` object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 18: Data model properties for `Computer Connection` object (alphabetical listing)

Property	Details
`Connection Name`	*Type:* `varchar`. Maximum: 64. Lists the current values of `connectionname` extracted from the compliance database using the following SQL query: `select ConnectionName as Value from ComplianceConnection order by ConnectionName`
`External ID`	*Type:* `int`. Maps to `compliancecomputerconnection.Externalid` in the compliance database.

Data model version 13.2

The Contract Object

This item maps to the `Contract` object in the FlexNet Manager Suite compliance database. It supports the addition of custom properties through the Business Importer.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the `Contract` appears in the `Contract` group in the right-click context menu.

Properties

The `Contract` object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 19: Data model properties for `Contract` object (alphabetical listing)

Property	Details
`Category ID`	*Type:* `varchar`. Maximum: 128. Stores the `categoryID` as a foreign key to the `category` table in the compliance database.
`Comments`	*Type:* `ntext`. Maps to `Contract.Comments` in the compliance database.
`Contract Description`	*Type:* `nvarchar`. Maximum: 100. Mandatory: adapters must provide values for this column. Maps to `Contract.ContractName` in the compliance database.
`Contract No`	*Type:* `nvarchar`. Maximum: 60. Mandatory: adapters must provide values for this column. Maps to `Contract.ContractNo` in the compliance database.
`Contract Status`	*Type:* `nvarchar`. Maximum: 1000. Values of `Contract Status` must match those stored in the `ContractStatus` table of the compliance database.
`Contract Type`	*Type:* `nvarchar`. Maximum: 1000. Values of `Contract Type` must match those stored in the `ContractType` table of the compliance database.
`Corporate Unit ID`	*Type:* `varchar`. Maximum: 128. Stores the `businessunitID` as a foreign key to the `CorporateUnit` table in the compliance database.
`Cost Center ID`	*Type:* `varchar`. Maximum: 128. Stores the `costcenterID` as a foreign key to the `costcenter` table in the compliance database.
`Evergreen`	*Type:* `bit`. Maps to `Contract.NeverExpires` in the compliance database.
`Expiry Date`	*Type:* `datetime`. Maps to `Contract.EndDate` in the compliance database.
`Global Amount`	*Type:* `money`. Maps to `Contract.TotalValue` in the compliance database.
`Global Amount Currency Rate ID`	*Type:* `int`. Stores the `TotalValueRateID` as a foreign key to the `currencyrate` table in the compliance database.

Property	Details
`Last Renewed Date`	*Type:* `datetime`. Maps to `Contract.LastRenewedDate` in the compliance database.
`Location ID`	*Type:* `varchar`. Maximum: 128. Stores the `locationID` as a foreign key to the `location` table in the compliance database.
`Master Contract ID`	*Type:* `int`. Stores the `MasterContractID` as a foreign key to the `Contract` table in the compliance database.
`Monthly Amount`	*Type:* `money`. Maps to `Contract.MonthlyValue` in the compliance database.
`Monthly Amount Currency Rate ID`	*Type:* `int`. Stores the `MonthlyValueRateID` as a foreign key to the `currencyrate` table in the compliance database.
`Next Renewal Date`	*Type:* `datetime`. Maps to `Contract.RenewalDate` in the compliance database.
`Previous Contract ID`	*Type:* `int`. Stores the `PreviousContractID` as a foreign key to the `Contract` table in the compliance database.
`Review Date`	*Type:* `datetime`. Maps to `Contract.PreExpiryDate` in the compliance database.
`Start Date`	*Type:* `datetime`. Maps to `Contract.StartDate` in the compliance database.
`Vendor ID`	*Type:* `int`. Stores the `VendorID` as a foreign key to the `Vendor` table in the compliance database.

Data model version 13.2

The Contract Responsibility Object

This item maps to the ComplianceResponsibility object in the FlexNet Manager Suite compliance database.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the Contract Responsibility appears in the Contract group in the right-click context menu.

Properties

The Contract Responsibility object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 20: Data model properties for Contract Responsibility object (alphabetical listing)

Property	Details
Comments	*Type:* ntext. Maps to ComplianceResponsibility.Comment in the compliance database.
Contract ID	Mandatory: adapters must provide values for this column. Stores the contractid as a foreign key to the Contract table in the compliance database.
Responsibility	*Type:* nvarchar. Maximum: 1000. Mandatory: adapters must provide values for this column. Values of Responsibility must match those stored in the ResponsibilityType table of the compliance database.
User ID	*Type:* int. Mandatory: adapters must provide values for this column. Stores the complianceuserid as a foreign key to the User table in the compliance database.

Data model version 13.2

The Corporate Unit Object

This item maps to the CorporateUnit object in the FlexNet Manager Suite compliance database.

Database interaction

If an incoming Corporate Unit is not matched by an existing record in the compliance database, a new record is created there.

If an incoming Corporate Unit matches an existing record in the compliance database, details are updated (as allowed for individual properties described below).

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the Corporate Unit appears in the Enterprise Groups group in the right-click context menu.

Properties

The Corporate Unit object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 21: Data model properties for Corporate Unit object (alphabetical listing)

Property	Details
Address - City	*Type:* varchar. Maximum: 200. Maps to CorporateUnit.address_city in the compliance database.
Address - Country	*Type:* varchar. Maximum: 100. Maps to CorporateUnit.address_country in the compliance database.
Address - State	*Type:* varchar. Maximum: 200. Maps to CorporateUnit.address_state in the compliance database.
Address - Street	*Type:* varchar. Maximum: 200. Maps to CorporateUnit.address_street in the compliance database.
Address - Zip	*Type:* varchar. Maximum: 20. Maps to CorporateUnit.address_zip in the compliance database.
Business Phone Number	*Type:* varchar. Maximum: 30. Maps to CorporateUnit.businessphonenumber in the compliance database.
Description	*Type:* ntext. Maps to CorporateUnit.comments in the compliance database.

Property	Details
`Email Address`	*Type:* `varchar`. Maximum: 200. Maps to `CorporateUnit.email` in the compliance database.
`Fax Number`	*Type:* `varchar`. Maximum: 30. Maps to `CorporateUnit.faxphonenumber` in the compliance database.
`ID`	*Type:* `int`. For the `ID` property, the Business Adapter Studio offers a list of the output field names from previous objects in the adapter. Maps to `CorporateUnit.groupexid` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. In the Business Adapter Studio, SQL-like expressions for pattern matching (contained in the `MatchingMask` attribute of the adapter's XML file) are supported for this property. An exception is the case when the incoming `ID` is null. In these cases, the property is excluded from the matching process, which then relies on other properties of the `Corporate Unit` object. However, updates to the stored value of `CorporateUnit.groupexid` are not permitted.
`Name`	*Type:* `varchar`. Maximum: 128. Mandatory: adapters must provide values for this column. Individual groups in the path through the enterprise group hierarchy are separated with the / character. Maps to `CorporateUnit.groupcn` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. However, updates to the stored value of `CorporateUnit.groupcn` are not permitted.

Data model version 13.2

The Cost Center Object

This item maps to the CostCenter object in the FlexNet Manager Suite compliance database.

Database interaction

If an incoming Cost Center is not matched by an existing record in the compliance database, a new record is created there.

If an incoming Cost Center matches an existing record in the compliance database, details are updated (as allowed for individual properties described below).

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the Cost Center appears in the Enterprise Groups group in the right-click context menu.

Within the Business Adapter Studio, the default name suggested for the Output attribute of this Cost Center object is CostCenter_ID.

Properties

The Cost Center object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 22: Data model properties for Cost Center object (alphabetical listing)

Property	Details
Address - City	*Type:* varchar. Maximum: 200. Maps to CostCenter.address_city in the compliance database.
Address - Country	*Type:* varchar. Maximum: 100. Maps to CostCenter.address_country in the compliance database.
Address - State	*Type:* varchar. Maximum: 200. Maps to CostCenter.address_state in the compliance database.
Address - Street	*Type:* varchar. Maximum: 200. Maps to CostCenter.address_street in the compliance database.
Address - Zip	*Type:* varchar. Maximum: 20. Maps to CostCenter.address_zip in the compliance database.
Business Phone Number	*Type:* varchar. Maximum: 30. Maps to CostCenter.businessphonenumber in the compliance database.

Property	Details
`Description`	*Type:* `ntext`. Maps to `CostCenter.comments` in the compliance database.
`Email Address`	*Type:* `varchar`. Maximum: 200. Maps to `CostCenter.email` in the compliance database.
`Fax Number`	*Type:* `varchar`. Maximum: 30. Maps to `CostCenter.faxphonenumber` in the compliance database.
`ID`	*Type:* `int`. For the `ID` property, the Business Adapter Studio offers a list of the output field names from previous objects in the adapter. Maps to `CostCenter.groupexid` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. In the Business Adapter Studio, SQL-like expressions for pattern matching (contained in the `MatchingMask` attribute of the adapter's XML file) are supported for this property. An exception is the case when the incoming `ID` is null. In these cases, the property is excluded from the matching process, which then relies on other properties of the `Cost Center` object. However, updates to the stored value of `CostCenter.groupexid` are not permitted.
`Name`	*Type:* `varchar`. Maximum: 128. Mandatory: adapters must provide values for this column. Individual groups in the path through the enterprise group hierarchy are separated with the / character. Maps to `CostCenter.groupcn` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. However, updates to the stored value of `CostCenter.groupcn` are not permitted.

Data model version 13.2

The Currency Object

This item maps to the currency object in the FlexNet Manager Suite compliance database.

Database interaction

***Note:** If an incoming Currency is not matched by an existing record in the compliance database, the incoming record is rejected. Creation of new records is not allowed for Currency objects.*

***Note:** If an incoming Currency matches an existing record in the compliance database, the incoming details are ignored. Updating existing records is not allowed for Currency objects.*

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the Currency appears in the Miscellaneous group in the right-click context menu.

Properties

The Currency object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 23: Data model properties for Currency object (alphabetical listing)

Property	Details
Currency Code	*Type:* nvarchar. Maximum: 32. Mandatory: adapters must provide values for this column. Values of Currency Code must match those stored in the CurrencyCode table of the compliance database.

Data model version 13.2

The Currency Rate Object

This item maps to the currencyrate object in the FlexNet Manager Suite compliance database.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the Currency Rate appears in the Miscellaneous group in the right-click context menu.

Properties

The Currency Rate object exposes the following properties in the data model for the Business Importer and the

Business Adapter Studio.

Table 24: Data model properties for `Currency Rate` object (alphabetical listing)

Property	Details
`Currency ID`	Mandatory: adapters must provide values for this column. Stores the `currencyid` as a foreign key to the `currency` table in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there.
`Exchange Rate`	*Type:* `float`. Mandatory: adapters must provide values for this column. Maps to `currencyrate.rate` in the compliance database.
`Snapshot ID`	Mandatory: adapters must provide values for this column. Stores the `snapshotid` as a foreign key to the `currencyratesnapshot` table in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there.

Data model version 13.2

The Currency Snapshot Object

This item maps to the `currencyratesnapshot` object in the FlexNet Manager Suite compliance database.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the `Currency Snapshot` appears in the `Miscellaneous` group in the right-click context menu.

Properties

The `Currency Snapshot` object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 25: Data model properties for `Currency Snapshot` object (alphabetical listing)

Property	Details
`Reference Currency ID`	In the Business Adapter Studio, the default value offered for `Reference Currency ID` is the current value of `DefaultCurrencyID` in the compliance database. Stores the `snapshotreferencecurrencyid` as a foreign key to the `currency` table in the compliance database.

Property	Details
`Snapshot Date`	*Type:* `datetime`. Mandatory: adapters must provide values for this column. Maps to `currencyratesnapshot.snapshotdate` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. Null values are included in the matching process, so that only corresponding records with a null or empty value for this property are matched.
`Snapshot Name`	Mandatory: adapters must provide values for this column. Maps to `currencyratesnapshot.snapshotname` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there.

Data model version 13.2

The Custom Query Object

The custom query object maps to the `Query` object in the compliance database, and does not contain any properties in this data model.

Data model version 13.2

The Document Object

This item maps to the `document` object in the FlexNet Manager Suite compliance database.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the `Document` appears in the `Miscellaneous` group in the right-click context menu.

Properties

The `Document` object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 26: Data model properties for `Document` object (alphabetical listing)

Property	Details
`Asset ID`	*Type:* `int`. Stores the `AssetID` as a foreign key to the `asset` table in the compliance database.

Property	Details
`Attached By`	*Type:* `nvarchar`. Maximum: 256. Maps to `document.UserName` in the compliance database.
`Contract ID`	*Type:* `int`. Stores the `contractid` as a foreign key to the `contract` table in the compliance database.
`Description`	*Type:* `nvarchar`. Maximum: 3000. Maps to `document.DocumentDescription` in the compliance database.
`Document Reference/Path`	*Type:* `nvarchar`. Maximum: 500. Mandatory: adapters must provide values for this column. Maps to `document.documentname` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there.
`Document Type ID`	*Type:* `int`. Mandatory: adapters must provide values for this column. Maps to `document.documenttypeid` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there.
`License ID`	*Type:* `int`. Stores the `softwarelicenseid` as a foreign key to the `License` table in the compliance database.
`Purchase Order ID`	*Type:* `int`. Stores the `purchaseorderid` as a foreign key to the `purchaseorder` table in the compliance database.
`User ID`	*Type:* `int`. Stores the `complianceuserid` as a foreign key to the `User` table in the compliance database.

Data model version 13.2

The Domain Object

This item maps to the `compliancedomain` object in the FlexNet Manager Suite compliance database.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the `Domain` appears in the `Miscellaneous` group in the right-click context menu.

Properties

The `Domain` object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 27: Data model properties for `Domain` object (alphabetical listing)

Property	Details
`Flat Name`	Mandatory: adapters must provide values for this column. In the Business Adapter Studio, the default regular expression to extract the `Flat Name` from the incoming data is `(?<=DC=.*DC=).*?(?=,DC=)` Maps to `compliancedomain.flatname` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there.

`Qualified Name`	Mandatory: adapters must provide values for this column.
	In the Business Adapter Studio, `Qualified Name` offers a default regular expression of
	`(?<=DC=).*`
	replacing incoming strings of
	`,DC=`
	with
	`.`
	Maps to `compliancedomain.QualifiedName` in the compliance database.
	However, updates to the stored value of `compliancedomain.QualifiedName` are not permitted.

Data model version 13.2

The License Object

This item maps to the License object in the FlexNet Manager Suite compliance database. It supports the addition of custom properties through the Business Importer.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the License appears in the License group in the right-click context menu.

Properties

The License object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 28: Data model properties for License object (alphabetical listing)

Property	Details
Access Granted to Users or Usage Consumes Entitlements	*Type:* bit. Maps to License.allowexternalroaminguse in the compliance database.
Additional External Users	*Type:* int. Maps to License.additionalbulkusersexternal in the compliance database.
Additional Infrequent Users	*Type:* int. Maps to License.additionalbulkusersinfrequent in the compliance database.
Additional Normal Users	*Type:* int. Maps to License.additionalbulkusersregular in the compliance database.
Allocations Consume License Entitlements	*Type:* bit. Maps to License.AlwaysInstalled in the compliance database.
Allow Sub-Capacity Licensing for Points outside ILMT	*Type:* bit. Maps to License.allowibmpvusubcapacityfromnonilmt in the compliance database.
Apply Special Virtualization Rights	*Type:* bit. Maps to License.CoverInstallsOnVirtualMachines in the compliance database.

Property	Details
`Apply User Limit Per Processor Core`	*Type:* `bit`. Maps to `License.minimumnumberofusersmultipliedbyprocessors` in the compliance database.
`Calculate Compliance`	*Type:* `bit`. Maps to `License.calculatecompliance` in the compliance database.
`Category ID`	*Type:* `varchar`. Maximum: 128. Stores the `categoryID` as a foreign key to the `category` table in the compliance database.
`Charges Amount`	*Type:* `float`. Maps to `License.ChargeBackPrice` in the compliance database.
`Charges Amount Currency Rate ID`	*Type:* `int`. Stores the `ChargeBackPriceRateID` as a foreign key to the `currencyrate` table in the compliance database.
`Charges Frequency`	*Type:* `nvarchar`. Maximum: 1000. Values of `Charges Frequency` must match those stored in the `PeriodType` table of the compliance database.
`Comments`	*Type:* `ntext`. Maps to `License.Comments` in the compliance database.
`Compliance Status`	*Type:* `nvarchar`. Maximum: 1000. Values of `Compliance Status` must match those stored in the `SoftwareLicenseComplianceStatus` table of the compliance database.
`Copy Version and Edition from Most Recent Application`	*Type:* `bit`. Maps to `License.copyeditionandversion` in the compliance database.
`Cores Limit`	*Type:* `int`. Maps to `License.numberofcores` in the compliance database.
`Corporate Unit ID`	*Type:* `varchar`. Maximum: 128. Stores the `businessunitID` as a foreign key to the `CorporateUnit` table in the compliance database.
`Cost Center ID`	*Type:* `varchar`. Maximum: 128. Stores the `costcenterID` as a foreign key to the `costcenter` table in the compliance database.

Property	Details
Count Includes Installations on Host OS	*Type:* bit. Maps to License.limitvirtualinstallsincludeshost in the compliance database.
Duration	*Type:* nvarchar. Maximum: 1000. Values of Duration must match those stored in the SoftwareLicenseDuration table of the compliance database.
Edition	*Type:* nvarchar. Maximum: 60. Maps to License.Edition in the compliance database.
Eligible for use with Cloud Service Providers	*Type:* bit. Maps to License.licensemobilityapplies in the compliance database.
Expiry Date	*Type:* datetime. Maps to License.ExpiryDate in the compliance database.
External User Multiplier	*Type:* float. Maps to License.usermultiplierexternal in the compliance database.
Extra Entitlements	*Type:* int. Maps to License.NumberPurchased in the compliance database.
Group Allocation Compliance Level	*Type:* int. Maps to License.groupallocationcompliancelevel in the compliance database.
Infrequent User Multiplier	*Type:* float. Maps to License.usermultiplierinfrequent in the compliance database.
License Key	*Type:* nvarchar. Maximum: 256. Maps to License.LicenseKey in the compliance database.
License Keys Rule	*Type:* nvarchar. Maximum: 1000. Values of License Keys Rule must match those stored in the SoftwareLicenseKeyType table of the compliance database.
License Status	*Type:* nvarchar. Maximum: 1000. Values of License Status must match those stored in the LicenseStatus table of the compliance database.
License Type	*Type:* nvarchar. Maximum: 1000. Values of License Type must match those stored in the SoftwareLicenseType table of the compliance database.

Property	Details
`License calculation uses host capacity`	*Type:* `bit`. Maps to `License.usehostprocessorinformation` in the compliance database.
`Location ID`	*Type:* `varchar`. Maximum: 128. Stores the `locationID` as a foreign key to the `location` table in the compliance database.
`Manage Linked Applications Automatically`	*Type:* `bit`. Maps to `License.automanagetitles` in the compliance database.
`Manager ID`	*Type:* `int`. Stores the `ManagerID` as a foreign key to the `user` table in the compliance database.
`Maximum Installations for a User`	*Type:* `int`. Maps to `License.numberofcomputersallowedperuserlicensepoint` in the compliance database.
`Maximum Installations on a Device`	*Type:* `int`. Maps to `License.numberofapplicationinstallsallowedperlicensepoint` in the compliance database.
`Maximum Sockets`	*Type:* `int`. Maps to `License.numberofsockets` in the compliance database.
`Maximum VMs Per License`	*Type:* `int`. Maps to `License.NumberOfAllowedVirtualInstalls` in the compliance database.
`Maximum Virtual Cores Per Instance`	*Type:* `int`. Maps to `License.totalnumberofcorespervmperlicense` in the compliance database.
`Maximum Virtual Processors Per License`	*Type:* `int`. Maps to `License.numberofprocessorsperose` in the compliance database.
`Minimum Core Licenses Per VM`	*Type:* `int`. Maps to `License.minimumnumberoflicensespervm` in the compliance database.
`Minimum Days before Reassignment`	*Type:* `int`. Maps to `License.reassignmenttimelimit` in the compliance database.

Property	Details
Minimum Processors	*Type:* int. Maps to License.minimumnumberofprocessors in the compliance database.
Minimum Users	*Type:* int. Maps to License.minimumnumberofusers in the compliance database.
Name	*Type:* nvarchar. Maximum: 256. Mandatory: adapters must provide values for this column. Maps to License.Name in the compliance database.
Number of Cores Per Socket License	*Type:* int. Maps to License.numberofcorespersocket in the compliance database.
Number of Processors Per License Consumed	*Type:* int. Maps to License.NumberOfAllowedProcessorsPerHost in the compliance database.
One Entitlement Allows User to Access Number of Installations	*Type:* bit. Maps to License.limitnumberofcomputersuserlicensecanbeinstalledon in the compliance database.
One Entitlement Covers Number of Installations on a Device	*Type:* bit. Maps to License.limitnumberofapplicationseachlicensepointcovers in the compliance database.
Override Unit Price	*Type:* float. Maps to License.PurchasePrice in the compliance database.
Override Unit Price Currency Rate ID	*Type:* int. Stores the PurchasePriceRateID as a foreign key to the currencyrate table in the compliance database.
PVU Entitlement Limit	*Type:* int. Maps to License.pvulimit in the compliance database.
PVU Limit Applies	*Type:* bit. Maps to License.pvulimitapplies in the compliance database.
Processors Limit	*Type:* int. Maps to License.numberofprocessors in the compliance database.
Publisher ID	*Type:* int. Stores the PublisherID as a foreign key to the vendor table in the compliance database.

Property	Details
`Resale Price`	*Type:* `float`. Maps to `License.ResalePrice` in the compliance database.
`Resale Price Currency Rate ID`	*Type:* `int`. Stores the `ResalePriceRateID` as a foreign key to the `currencyrate` table in the compliance database.
`Resale Recipient`	*Type:* `nvarchar`. Maximum: 128. Maps to `License.EndOfLifeRecipient` in the compliance database.
`Resources Consumed`	*Type:* `int`. Maps to `License.resourceunitsconsumed` in the compliance database.
`Restrict the number of VMs per licensed host`	*Type:* `bit`. Maps to `License.LimitNumberOfVirtualInstalls` in the compliance database.
`Retirement Date`	*Type:* `datetime`. Maps to `License.RetirementDate` in the compliance database.
`Retirement Reason`	*Type:* `varchar`. Maximum: 1000. Values of `Retirement Reason` must match those stored in the `EndOfLifeReason` table of the compliance database.
`Second Use At Home`	*Type:* `bit`. Maps to `License.SecondUsageAtHome` in the compliance database.
`Second Use On Work Laptop`	*Type:* `bit`. Maps to `License.SecondUsageWorkLaptop` in the compliance database.
`Subject To True Up`	*Type:* `bit`. Maps to `License.TrueUp` in the compliance database.
`Third Party Users may use Linked Applications`	*Type:* `bit`. Maps to `License.thirdpartyaccessallowed` in the compliance database.
`Time Limit Applies before License Reassignment`	*Type:* `bit`. Maps to `License.reassignmenttimelimitapplies` in the compliance database.
`Unlimited Consumption`	*Type:* `bit`. Maps to `License.UnlimitedConsumption` in the compliance database.

Property	Details
`Usage Time Limit In Days`	*Type:* `int`. Maps to `License.virtualapplicationaccessmaximumusageperiod` in the compliance database.
`Version`	*Type:* `nvarchar`. Maximum: 60. Mandatory: adapters must provide values for this column. Maps to `License.Version` in the compliance database.

Data model version 13.2

The Link Contract - Asset Object

This item maps to the `contractasset` object in the FlexNet Manager Suite compliance database.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the `Link Contract - Asset` appears in the `Relationships` group in the right-click context menu.

Properties

The `Link Contract - Asset` object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 29: Data model properties for `Link Contract - Asset` object (alphabetical listing)

Property	Details
`Asset ID`	Stores the `assetid` as a foreign key to the `Asset` table in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. However, updates to the stored value of `contractasset.assetid` are not permitted.
`Contract ID`	Stores the `contractid` as a foreign key to the `Contract` table in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. However, updates to the stored value of `contractasset.contractid` are not permitted.

Data model version 13.2

The Link Contract - License Object

This item maps to the contractlicense object in the FlexNet Manager Suite compliance database.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the Link Contract - License appears in the Relationships group in the right-click context menu.

Properties

The Link Contract - License object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 30: Data model properties for Link Contract - License object (alphabetical listing)

Property	Details
Contract ID	Stores the contractid as a foreign key to the Contract table in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. However, updates to the stored value of contractlicense.contractid are not permitted.
License ID	Stores the softwarelicenseid as a foreign key to the License table in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. However, updates to the stored value of contractlicense.softwarelicenseid are not permitted.

Data model version 13.2

The Link Operator - Role Object

This item maps to the ComplianceOperatorRole object in the FlexNet Manager Suite compliance database.

Database interaction

***Note:** If an incoming Link Operator - Role matches an existing record in the compliance database, the incoming details are ignored. Updating existing records is not allowed for Link Operator - Role objects.*

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the `Link Operator - Role` appears in the `Operator` group in the right-click context menu.

Properties

The `Link Operator - Role` object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 31: Data model properties for `Link Operator - Role` object (alphabetical listing)

Property	Details
`Operator ID`	Stores the `targetid` as a foreign key to the `ComplianceOperator` table in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. However, updates to the stored value of `ComplianceOperatorRole.targetid` are not permitted.
`Role`	*Type:* `nvarchar`. Maximum: 64. Values of `Role` must match those stored in the `ComplianceRole` table of the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. However, updates to the stored value of `ComplianceOperatorRole.compliancerole` are not permitted.

Data model version 13.2

The Link Payment Schedule - Asset Object

This item maps to the `paymentscheduleasset` object in the FlexNet Manager Suite compliance database.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the `Link Payment Schedule - Asset` appears in the `Payment Schedule` group in the right-click context menu.

Properties

The `Link Payment Schedule - Asset` object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 32: Data model properties for Link Payment Schedule - Asset object (alphabetical listing)

Property	Details
Asset ID	Stores the assetid as a foreign key to the asset table in the compliance database. However, updates to the stored value of paymentscheduleasset.assetid are not permitted.
Contract ID	Stores the contractid as a foreign key to the contract table in the compliance database. However, updates to the stored value of paymentscheduleasset.contractid are not permitted.
Coverage End Date	*Type:* datetime. Maps to paymentscheduleasset.ActiveEndDate in the compliance database.
Coverage Start Date	*Type:* datetime. Maps to paymentscheduleasset.ActiveStartDate in the compliance database.
Payment Schedule ID	*Type:* int. Stores the PaymentScheduleID as a foreign key to the PaymentSchedule table in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. However, updates to the stored value of paymentscheduleasset.PaymentScheduleID are not permitted.

Data model version 13.2

The Link Payment Schedule - License Object

This item maps to the paymentschedulelicense object in the FlexNet Manager Suite compliance database.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the Link Payment Schedule - License appears in the Payment Schedule group in the right-click context menu.

Properties

The Link Payment Schedule - License object exposes the following properties in the data model for the

Business Importer and the Business Adapter Studio.

Table 33: Data model properties for `Link Payment Schedule - License` object (alphabetical listing)

Property	Details
`Contract ID`	Stores the `contractid` as a foreign key to the `contract` table in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. However, updates to the stored value of `paymentschedulelicense.contractid` are not permitted.
`License ID`	Stores the `softwarelicenseid` as a foreign key to the `License` table in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. However, updates to the stored value of `paymentschedulelicense.softwarelicenseid` are not permitted.
`Payment Schedule ID`	*Type:* `int`. Stores the `PaymentScheduleID` as a foreign key to the `PaymentSchedule` table in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. However, updates to the stored value of `paymentschedulelicense.PaymentScheduleID` are not permitted.

Data model version 13.2

The Link Purchase Order Line - Asset Object

This item maps to the `purchaseorderlineasset` object in the FlexNet Manager Suite compliance database.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the `Link Purchase Order Line - Asset` appears in the `Relationships` group in the right-click context menu.

Properties

The `Link Purchase Order Line - Asset` object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 34: Data model properties for `Link Purchase Order Line - Asset` object (alphabetical listing)

Property	Details
`Asset ID`	Stores the `assetid` as a foreign key to the `Asset` table in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. However, updates to the stored value of `purchaseorderlineasset.assetid` are not permitted.
`Purchase Order Line ID`	Stores the `purchaseorderdetailid` as a foreign key to the `PurchaseOrderLine` table in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. However, updates to the stored value of `purchaseorderlineasset.purchaseorderdetailid` are not permitted.

Data model version 13.2

The Link Purchase Order Line - License Object

This item maps to the `purchaseorderlinelicense` object in the FlexNet Manager Suite compliance database.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the `Link Purchase Order Line - License` appears in the `Relationships` group in the right-click context menu.

Properties

The `Link Purchase Order Line - License` object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 35: Data model properties for `Link Purchase Order Line - License` object (alphabetical listing)

Property	Details
`License ID`	Stores the `softwarelicenseid` as a foreign key to the `License` table in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there.

Property	Details
Purchase Order Line ID	Stores the purchaseorderdetailid as a foreign key to the PurchaseOrderLine table in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there.

Data model version 13.2

The Location Object

This item maps to the Location object in the FlexNet Manager Suite compliance database.

Database interaction

If an incoming Location is not matched by an existing record in the compliance database, a new record is created there.

If an incoming Location matches an existing record in the compliance database, details are updated (as allowed for individual properties described below).

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the Location appears in the Enterprise Groups group in the right-click context menu.

Within the Business Adapter Studio, the default name suggested for the Output attribute of this Location object is Location_ID.

Properties

The Location object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 36: Data model properties for Location object (alphabetical listing)

Property	Details
Address - City	*Type:* varchar. Maximum: 200. Maps to Location.address_city in the compliance database.
Address - Country	*Type:* varchar. Maximum: 100. Maps to Location.address_country in the compliance database.
Address - State	*Type:* varchar. Maximum: 200. Maps to Location.address_state in the compliance database.

Property	Details
`Address - Street`	*Type:* `varchar`. Maximum: 200. Maps to `Location.address_street` in the compliance database.
`Address - Zip`	*Type:* `varchar`. Maximum: 20. Maps to `Location.address_zip` in the compliance database.
`Business Phone Number`	*Type:* `varchar`. Maximum: 30. Maps to `Location.businessphonenumber` in the compliance database.
`Description`	*Type:* `ntext`. Maps to `Location.comments` in the compliance database.
`Email Address`	*Type:* `varchar`. Maximum: 200. Maps to `Location.email` in the compliance database.
`Fax Number`	*Type:* `varchar`. Maximum: 30. Maps to `Location.faxphonenumber` in the compliance database.
`ID`	*Type:* `int`. For the `ID` property, the Business Adapter Studio offers a list of the output field names from previous objects in the adapter. Maps to `Location.groupexid` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. In the Business Adapter Studio, SQL-like expressions for pattern matching (contained in the `MatchingMask` attribute of the adapter's XML file) are supported for this property. An exception is the case when the incoming `ID` is null. In these cases, the property is excluded from the matching process, which then relies on other properties of the `Location` object. However, updates to the stored value of `Location.groupexid` are not permitted.
`Name`	*Type:* `varchar`. Maximum: 128. Mandatory: adapters must provide values for this column. Individual groups in the path through the enterprise group hierarchy are separated with the / character. Maps to `Location.groupcn` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there. However, updates to the stored value of `Location.groupcn` are not permitted.

Data model version 13.2

The Operator Object

This item maps to the ComplianceOperator object in the FlexNet Manager Suite compliance database.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the Operator appears in the Operator group in the right-click context menu.

Properties

The Operator object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 37: Data model properties for Operator object (alphabetical listing)

Property	Details
Account	*Type:* nvarchar. Maximum: 256. Mandatory: adapters must provide values for this column. Maps to ComplianceOperator.operatorlogin in the compliance database.
Email	*Type:* nvarchar. Maximum: 200. Maps to ComplianceOperator.email in the compliance database.
Enabled	*Type:* bool. Mandatory: adapters must provide values for this column. Maps to ComplianceOperator.isenabled in the compliance database.
Job Title	*Type:* nvarchar. Maximum: 128. Maps to ComplianceOperator.JobTitle in the compliance database.
Name	*Type:* nvarchar. Maximum: 512. Maps to ComplianceOperator.operatorname in the compliance database.

Data model version 13.2

The Payment Schedule Object

This item maps to the paymentschedule object in the FlexNet Manager Suite compliance database.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the Payment Schedule appears in the Payment Schedule group in the right-click context menu.

Properties

The Payment Schedule object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 38: Data model properties for Payment Schedule object (alphabetical listing)

Property	Details
Buyout Cost	*Type:* money. Maps to paymentschedule.BuyoutCost in the compliance database.
Buyout Cost Currency Rate ID	*Type:* int. Stores the BuyoutCostRateID as a foreign key to the currencyrate table in the compliance database.
Contract ID	*Type:* int. Mandatory: adapters must provide values for this column. Stores the contractid as a foreign key to the contract table in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there.
End Date	*Type:* datetime. Maps to paymentschedule.enddate in the compliance database.
Frequency	*Type:* nvarchar. Maximum: 1000. Values of Frequency must match those stored in the PeriodType table of the compliance database.
Lease Number	*Type:* nvarchar. Maximum: 150. Maps to paymentschedule.LeaseNumber in the compliance database.
Lease Termination Date	*Type:* datetime. Maps to paymentschedule.LeaseTerminationDate in the compliance database.

Property	Details
`Lease Termination Reason`	*Type:* `nvarchar`. Maximum: 100. Maps to `paymentschedule.LeaseTerminationReason` in the compliance database.
`Link to New Assets and Licenses attached to Contract`	*Type:* `bit`. Mandatory: adapters must provide values for this column. Maps to `paymentschedule.includenewassetsandlicenses` in the compliance database.
`Name`	*Type:* `nvarchar`. Maximum: 100. Mandatory: adapters must provide values for this column. Maps to `paymentschedule.description` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there.
`Notes`	*Type:* `ntext`. Maps to `paymentschedule.Comment` in the compliance database.
`Payment Schedule Metric`	Values of `Payment Schedule Metric` must match those stored in the `PaymentScheduleCategory` table of the compliance database.
`Payment Schedule Term`	Values of `Payment Schedule Term` must match those stored in the `PaymentScheduleTerm` table of the compliance database.
`Payment Schedule Type`	Values of `Payment Schedule Type` must match those stored in the `PaymentScheduleType` table of the compliance database.
`Start Date`	*Type:* `datetime`. Mandatory: adapters must provide values for this column. Maps to `paymentschedule.startdate` in the compliance database.

Data model version 13.2

The Payment Schedule Detail Object

This item maps to the `PaymentScheduleDetail` object in the FlexNet Manager Suite compliance database.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the `Payment Schedule Detail` appears in the `Payment Schedule` group in the right-click context menu.

Properties

The `Payment Schedule Detail` object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 39: Data model properties for `Payment Schedule Detail` object (alphabetical listing)

Property	Details
`Actual/Fixed Amount`	*Type:* `money`. Maps to `PaymentScheduleDetail.ActualAmount` in the compliance database.
`Actual/Fixed Amount Currency Rate ID`	*Type:* `int`. Stores the `ActualAmountRateID` as a foreign key to the `currencyrate` table in the compliance database.
`Budgeted Amount`	*Type:* `money`. Maps to `PaymentScheduleDetail.BudgetedAmount` in the compliance database.
`Budgeted Amount Currency Rate ID`	*Type:* `int`. Stores the `BudgetedAmountRateID` as a foreign key to the `currencyrate` table in the compliance database.
`Estimated Amount`	*Type:* `money`. Maps to `PaymentScheduleDetail.EstimatedAmount` in the compliance database.
`Estimated Amount Currency Rate ID`	*Type:* `int`. Stores the `EstimatedAmountRateID` as a foreign key to the `currencyrate` table in the compliance database.
`Installment Covered`	*Type:* `nvarchar`. Maximum: 50. Maps to `PaymentScheduleDetail.PeriodCovered` in the compliance database.
`Installment End Date`	*Type:* `datetime`. Mandatory: adapters must provide values for this column. Maps to `PaymentScheduleDetail.PeriodEndDate` in the compliance database.
`Installment Start Date`	*Type:* `datetime`. Mandatory: adapters must provide values for this column. Maps to `PaymentScheduleDetail.PeriodStartDate` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there.
`Installment Status`	*Type:* `int`. Values of `Installment Status` must match those stored in the `PaymentScheduledetailPaymentStatus` table of the compliance database.

Property	Details
Notes	*Type:* ntext. Maps to PaymentScheduleDetail.Notes in the compliance database.
Obligated to Pay	*Type:* bit. Maps to PaymentScheduleDetail.Obligated in the compliance database.
Payment Date	*Type:* datetime. Maps to PaymentScheduleDetail.PaymentDate in the compliance database.
Payment Schedule ID	*Type:* int. Mandatory: adapters must provide values for this column. Stores the PaymentScheduleID as a foreign key to the PaymentSchedule table in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there.
Quantity	*Type:* int. Maps to PaymentScheduleDetail.Quantity in the compliance database.
Software Assurance Unit Price	*Type:* money. Maps to PaymentScheduleDetail.SoftwareAssuranceUnitPrice in the compliance database.
Software Assurance Unit Price Rate ID	*Type:* int. Stores the SoftwareAssuranceUnitPriceRateID as a foreign key to the currencyrate table in the compliance database.
Unit Price	*Type:* money. Maps to PaymentScheduleDetail.UnitPrice in the compliance database.
Unit Price Currency Rate ID	*Type:* int. Stores the UnitPriceRateID as a foreign key to the currencyrate table in the compliance database.

Data model version 13.2

The Purchase Order Object

This item maps to the PurchaseOrder object in the FlexNet Manager Suite compliance database. It supports the addition of custom properties through the Business Importer.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the Purchase Order appears in the Purchase Order group in the right-click context menu.

Properties

The Purchase Order object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 40: Data model properties for Purchase Order object (alphabetical listing)

Property	Details
Purchase Date	*Type:* datetime. Mandatory: adapters must provide values for this column. Maps to PurchaseOrder.PurchaseOrderDate in the compliance database.
Purchase Order No	*Type:* nvarchar. Maximum: 50. Mandatory: adapters must provide values for this column. Maps to PurchaseOrder.PurchaseOrderNo in the compliance database.
Vendor ID	*Type:* int. Stores the VendorID as a foreign key to the Vendor table in the compliance database.

Data model version 13.2

The Purchase Order Line Object

This item maps to the PurchaseOrderLine object in the FlexNet Manager Suite compliance database. It supports the addition of custom properties through the Business Importer.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the Purchase Order Line appears in the Purchase Order group in the right-click context menu.

Properties

The Purchase Order Line object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 41: Data model properties for Purchase Order Line object (alphabetical listing)

Property	Details
Authorized By ID	*Type:* int. Stores the AuthorizedByID as a foreign key to the User table in the compliance database.
Category ID	*Type:* varchar. Maximum: 128. Stores the categoryID as a foreign key to the category table in the compliance database.
Comments	*Type:* ntext. Maps to PurchaseOrderLine.Comments in the compliance database.
Contract ID	*Type:* int. Stores the ContractID as a foreign key to the Contract table in the compliance database.
Corporate Unit ID	*Type:* varchar. Maximum: 128. Stores the businessunitID as a foreign key to the CorporateUnit table in the compliance database.
Cost Center ID	*Type:* varchar. Maximum: 128. Stores the costcenterID as a foreign key to the costcenter table in the compliance database.
Description	*Type:* nvarchar. Maximum: 250. Mandatory: adapters must provide values for this column. Maps to PurchaseOrderLine.ItemDescription in the compliance database.
Effective Date	*Type:* datetime. Maps to PurchaseOrderLine.EffectiveDate in the compliance database.
Expiry Date	*Type:* datetime. Maps to PurchaseOrderLine.ExpiryDate in the compliance database.
Include Support, Maintenance or Service Agreement	*Type:* bit. Maps to PurchaseOrderLine.MaintenanceOrServiceAgreement in the compliance database.
Invoice Date	*Type:* datetime. Maps to PurchaseOrderLine.InvoiceDate in the compliance database.
Invoice Number	*Type:* nvarchar. Maximum: 50. Maps to PurchaseOrderLine.InvoiceNo in the compliance database.

Property	Details
`Location ID`	*Type:* `varchar`. Maximum: 128. Stores the `locationID` as a foreign key to the `location` table in the compliance database.
`Part No/SKU`	*Type:* `nvarchar`. Maximum: 100. Maps to `PurchaseOrderLine.LicensePartNo` in the compliance database.
`Processed By ID`	*Type:* `int`. Stores the `ProcessedByID` as a foreign key to the `User` table in the compliance database.
`Publisher ID`	*Type:* `int`. Stores the `PublisherID` as a foreign key to the `vendor` table in the compliance database.
`Purchase Order ID`	*Type:* `int`. Mandatory: adapters must provide values for this column. Stores the `PurchaseOrderID` as a foreign key to the `PurchaseOrder` table in the compliance database.
`Purchase Order Line Sequence`	*Type:* `int`. Mandatory: adapters must provide values for this column. Maps to `PurchaseOrderLine.SequenceNumber` in the compliance database.
`Purchase Quantity`	*Type:* `int`. Maps to `PurchaseOrderLine.Quantity` in the compliance database.
`Purchase Type`	*Type:* `nvarchar`. Values of `Purchase Type` must match those stored in the `PurchaseOrderDetailType` table of the compliance database.
`Quantity Per Unit`	*Type:* `int`. Maps to `PurchaseOrderLine.quantityperunit` in the compliance database.
`Request Date`	*Type:* `datetime`. Maps to `PurchaseOrderLine.RequestDate` in the compliance database.
`Request Number`	*Type:* `nvarchar`. Maximum: 120. Maps to `PurchaseOrderLine.RequestNo` in the compliance database.
`Requestor ID`	*Type:* `int`. Stores the `RequestedByID` as a foreign key to the `User` table in the compliance database.
`Sales Tax`	*Type:* `float`. Maps to `PurchaseOrderLine.SalesTax` in the compliance database.

Property	Details
`Sales Tax Currency Rate ID`	*Type:* `int`. Stores the `SalesTaxRateID` as a foreign key to the `currencyrate` table in the compliance database.
`Shipping And Handling Currency Rate ID`	*Type:* `int`. Stores the `ShippingAndHandlingRateID` as a foreign key to the `currencyrate` table in the compliance database.
`Shipping Date`	*Type:* `datetime`. Maps to `PurchaseOrderLine.ShippingDate` in the compliance database.
`Shipping Location ID`	*Type:* `varchar`. Maximum: 128. Stores the `ShippingLocationID` as a foreign key to the `location` table in the compliance database.
`Shipping and Handling`	*Type:* `float`. Maps to `PurchaseOrderLine.ShippingAndHandling` in the compliance database.
`Status`	*Type:* `nvarchar`. Values of `Status` must match those stored in the `PurchaseOrderDetailStatus` table of the compliance database.
`Total Price Currency Rate ID`	*Type:* `int`. Stores the `TotalPriceRateID` as a foreign key to the `currencyrate` table in the compliance database.
`Unit Price`	*Type:* `float`. Maps to `PurchaseOrderLine.UnitPrice` in the compliance database.
`Unit Price Currency Rate ID`	*Type:* `int`. Stores the `UnitPriceRateID` as a foreign key to the `currencyrate` table in the compliance database.

Data model version 13.2

The Software Allocation Object

This item maps to the SoftwareLicenseAllocation object in the FlexNet Manager Suite compliance database.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the Software Allocation appears in the License group in the right-click context menu.

Properties

The Software Allocation object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 42: Data model properties for Software Allocation object (alphabetical listing)

Property	Details
Computer ID	*Type:* int. Stores the compliancecomputerid as a foreign key to the Computer table in the compliance database.
Exemption Reason	*Type:* nvarchar. Maximum: 1000. Values of Exemption Reason must match those stored in the SoftwareLicenseExemptionReason table of the compliance database.
License ID	*Type:* int. Mandatory: adapters must provide values for this column. Stores the softwarelicenseid as a foreign key to the License table in the compliance database.
License Key	*Type:* nvarchar. Maximum: 400. Maps to SoftwareLicenseAllocation.keyvalue in the compliance database.
Number allocated	*Type:* int. Maps to SoftwareLicenseAllocation.numberallocated in the compliance database.
Status	*Type:* nvarchar. Maximum: 1000. Values of Status must match those stored in the SoftwareLicenseAllocationstatus table of the compliance database.
User ID	*Type:* int. Mandatory: adapters must provide values for this column. Stores the complianceuserid as a foreign key to the User table in the compliance database.

Data model version 13.2

The Software License Key Object

This item maps to the SoftwareLicenseKey object in the FlexNet Manager Suite compliance database.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the Software License Key appears in the License group in the right-click context menu.

Properties

The Software License Key object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 43: Data model properties for Software License Key object (alphabetical listing)

Property	Details
License ID	Mandatory: adapters must provide values for this column. Stores the softwarelicenseid as a foreign key to the License table in the compliance database.
License Key	*Type:* varchar. Maximum: 400. Mandatory: adapters must provide values for this column. Maps to SoftwareLicenseKey.keyvalue in the compliance database.

Data model version 13.2

The Terms And Conditions Object

This item maps to the TermAndCondition object in the FlexNet Manager Suite compliance database.

Behavior in the Business Adapter Studio

When you create new objects in the Business Adapter Studio, the Terms And Conditions appears in the Contract group in the right-click context menu.

Properties

The Terms And Conditions object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 44: Data model properties for `Terms And Conditions` object (alphabetical listing)

Property	Details
`Begin Date`	*Type:* `datetime`. Maps to `TermAndCondition.begindate` in the compliance database.
`Comments`	*Type:* `ntext`. Maps to `TermAndCondition.comments` in the compliance database.
`Contract ID`	*Type:* `int`. Mandatory: adapters must provide values for this column. Maps to `TermAndCondition.contractid` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there.
`Description`	*Type:* `nvarchar`. Maximum: 100. Mandatory: adapters must provide values for this column. Maps to `TermAndCondition.description` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there.
`Document Reference`	*Type:* `nvarchar`. Maximum: 100. Maps to `TermAndCondition.docreference` in the compliance database.
`End Date`	*Type:* `datetime`. Maps to `TermAndCondition.enddate` in the compliance database.
`Term and Condition Type`	*Type:* `nvarchar`. Maximum: 100. Maps to `TermAndCondition.termandconditiontype` in the compliance database. It is used (perhaps in conjunction with other properties) as a key for matching existing records there.

Data model version 13.2

The User Object

This item maps to the User object in the FlexNet Manager Suite compliance database. It supports the addition of custom properties through the Business Importer.

Properties

The User object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 45: Data model properties for User object (alphabetical listing)

Property	Details
Account Name	*Type:* nvarchar. Maximum: 64. Maps to User.SAMAccountName in the compliance database.
Address - City	*Type:* nvarchar. Maximum: 200. Maps to User.Address_City in the compliance database.
Address - Country	*Type:* nvarchar. Maximum: 100. Maps to User.Address_Country in the compliance database.
Address - State	*Type:* nvarchar. Maximum: 200. Maps to User.Address_State in the compliance database.
Address - Street	*Type:* nvarchar. Maximum: 200. Maps to User.Address_Street in the compliance database.
Address - Zip	*Type:* nvarchar. Maximum: 20. Maps to User.Address_ZIP in the compliance database.
AlternateEmail	*Type:* nvarchar. Maximum: 200. Maps to User.AlternateEmail in the compliance database.
Category ID	*Type:* varchar. Maximum: 128. Stores the categoryID as a foreign key to the category table in the compliance database.
Corporate Unit ID	*Type:* varchar. Maximum: 128. Stores the businessunitID as a foreign key to the CorporateUnit table in the compliance database.
Cost Center ID	*Type:* varchar. Maximum: 128. Stores the costcenterID as a foreign key to the costcenter table in the compliance database.

Property	Details
`Domain ID`	*Type:* `int`. Stores the `ComplianceDomainID` as a foreign key to the `ComplianceDomain` table in the compliance database.
`Email`	*Type:* `nvarchar`. Maximum: 200. Maps to `User.Email` in the compliance database.
`Employee Number`	*Type:* `nvarchar`. Maximum: 128. Maps to `User.EmployeeNumber` in the compliance database.
`Employment Status`	*Type:* `nvarchar`. Maximum: 1000. Values of `Employment Status` must match those stored in the `EmploymentStatus` table of the compliance database.
`Fax Number`	*Type:* `nvarchar`. Maximum: 30. Maps to `User.FaxPhoneNumber` in the compliance database.
`First Name`	*Type:* `nvarchar`. Maximum: 128. Maps to `User.FirstName` in the compliance database.
`Inventory Source`	*Type:* `nvarchar`. Maximum: 64. Maps to `User.InventoryAgent` in the compliance database.
`Job Title`	*Type:* `nvarchar`. Maximum: 128. Maps to `User.JobTitle` in the compliance database.
`Last Name`	*Type:* `nvarchar`. Maximum: 128. Maps to `User.LastName` in the compliance database.
`Location ID`	*Type:* `varchar`. Maximum: 128. Stores the `locationID` as a foreign key to the `location` table in the compliance database.
`Manager ID`	*Type:* `int`. Stores the `ManagerID` as a foreign key to the `user` table in the compliance database.
`Middle Name`	*Type:* `nvarchar`. Maximum: 128. Maps to `User.MiddleName` in the compliance database.
`Mobile Number`	*Type:* `nvarchar`. Maximum: 30. Maps to `User.MobilePhoneNumber` in the compliance database.
`Phone Number`	*Type:* `nvarchar`. Maximum: 30. Maps to `User.BusinessPhoneNumber` in the compliance database.

Property	Details
User Full Name	*Type:* nvarchar. Maximum: 512. Maps to User.UserName in the compliance database.
User Status	*Type:* nvarchar. Maximum: 1000. Values of User Status must match those stored in the ComplianceUserStatus table of the compliance database.
User Suffix	*Type:* nvarchar. Maximum: 1000. Values of User Suffix must match those stored in the UserSuffix table of the compliance database.
User Title	*Type:* nvarchar. Maximum: 1000. Values of User Title must match those stored in the UserTitle table of the compliance database.

Data model version 13.2

The Vendor Object

This item maps to the Vendor object in the FlexNet Manager Suite compliance database. It supports the addition of custom properties through the Business Importer.

Properties

The Vendor object exposes the following properties in the data model for the Business Importer and the Business Adapter Studio.

Table 46: Data model properties for Vendor object (alphabetical listing)

Property	Details
Address - City	*Type:* nvarchar. Maximum: 200. Maps to Vendor.Address_City in the compliance database.
Address - Country	*Type:* nvarchar. Maximum: 100. Maps to Vendor.Address_Country in the compliance database.
Address - State	*Type:* nvarchar. Maximum: 200. Maps to Vendor.Address_State in the compliance database.
Address - Street	*Type:* nvarchar. Maximum: 200. Maps to Vendor.Address_Street in the compliance database.

Property	Details
`Address - Zip`	*Type:* `nvarchar`. Maximum: 20. Maps to `Vendor.Address_ZIP` in the compliance database.
`Email`	*Type:* `nvarchar`. Maximum: 200. Maps to `Vendor.Email` in the compliance database.
`Fax Number`	*Type:* `nvarchar`. Maximum: 30. Maps to `Vendor.FaxPhoneNumber` in the compliance database.
`Name`	*Type:* `nvarchar`. Maximum: 64. Mandatory: adapters must provide values for this column. Maps to `Vendor.VendorName` in the compliance database.
`Parent Vendor ID`	*Type:* `int`. Stores the `ParentVendorID` as a foreign key to the `Vendor` table in the compliance database.
`Phone Number`	*Type:* `nvarchar`. Maximum: 30. Maps to `Vendor.BusinessPhoneNumber` in the compliance database.
`Website`	*Type:* `nvarchar`. Maximum: 200. Maps to `Vendor.WebSite` in the compliance database.

Data model version 13.2

II

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 part introduces both business adapters, and the Business Adapter Studio that you can use to custom-build them. A completed business adapter can then be exercised by the Business Importer, as documented in the previous part.

1

Introducing the Business Adapter Studio

What is a business adapter?

A business adapter is an XML file that:

- Defines a connection to a data source (which may be a database system within your enterprise infrastructure, or other things including a well-formed spreadsheet)
- Maps the columns from the data source to a standard set of objects and attributes that can be imported into the operations database for FlexNet Manager Suite.

Examples of business information that may be relevant to your software and hardware asset management include:

- Details of your organization structure (to form enterprise groups in FlexNet Manager Suite)
- Purchase orders (especially relating to software purchases, upgrades, and maintenance)
- Contract details

and the like.

How is a business adapter used?

Business adapters may be used in two modes:

- Connected mode where the adapter is run on your central application server with direct connection to your operations databases.
- Disconnected mode where the business adapter runs on an inventory beacon, and cannot directly access the operations databases. This mode requires tighter security, and regular uploads of archived business information are automatically uploaded to the application server, and processed in a separate stage that is not controlled by the business adapter.

In connected mode, a Windows scheduled task on the application server triggers the Business Importer to read the business adapter. In disconnected mode, running the adapter is separate from importing the results. Triggered by the inventory beacon on a daily schedule you specify, the business adapter is read by the Business Importer, which then

- Connects to the specified connection
- Gathers the data defined by the XML in the adapter
- Collects the results into an archive package on the inventory beacon in disconnected mode; or writes the data into staging tables in connected mode
- Immediately uploads the package to the operations database in disconnected mode; or processes the data from the staging tables into the operations databases in connected mode
- Repeats this process for each of the other currently enabled adapters awaiting execution.

Therefore in connected mode, the data is available in the web interface within a very short processing time; but disconnected mode takes a little longer. The uploaded packages are held in a staging directory until all previous imports from business adapters are completed, and then the new arrival is processed into the operations database. Thereafter the business information is available in the web interface.

How is a business adapter created and maintained?

Business adapters are edited in the Business Adapter Studio. In this tool, you can:

- Develop the adapter in a protected, test environment, starting with templates that help keep your adapter compliant with the requirements of the central operations databases
- In connected mode (on your application server), simulate running the adapter to test and trouble-shoot its development
- Move the complete adapter into production.

Separately for business adapters running in disconnected mode, in the interface for the inventory beacon, you can turn any production-ready business adapter on or off (enabled/disabled), and schedule the time of day when all your enabled business adapters are run. As already noted, in connected mode, you use a Windows scheduled task to control timing.

Prerequisites

In brief:

- **System requirements:** Included in the requirements for the inventory beacon (for disconnected mode). The release notes for the inventory beacon are available from the download center on the Flexera website. The URL for this download location was provided in the e-mail you received from Flexera order processing, confirming your purchased products. The same website also contains the release notes for FlexNet Manager Suite, which include requirements for the Business Adapter Studio on the application server if you are preparing business adapters to run in connected mode.

  ***Tip:** If your proposed business adapter is to import any spreadsheet data in the `.xlsx` file format, you must first install a 32-bit version of Microsoft Access Database Engine on the inventory beacon that will perform this import.*

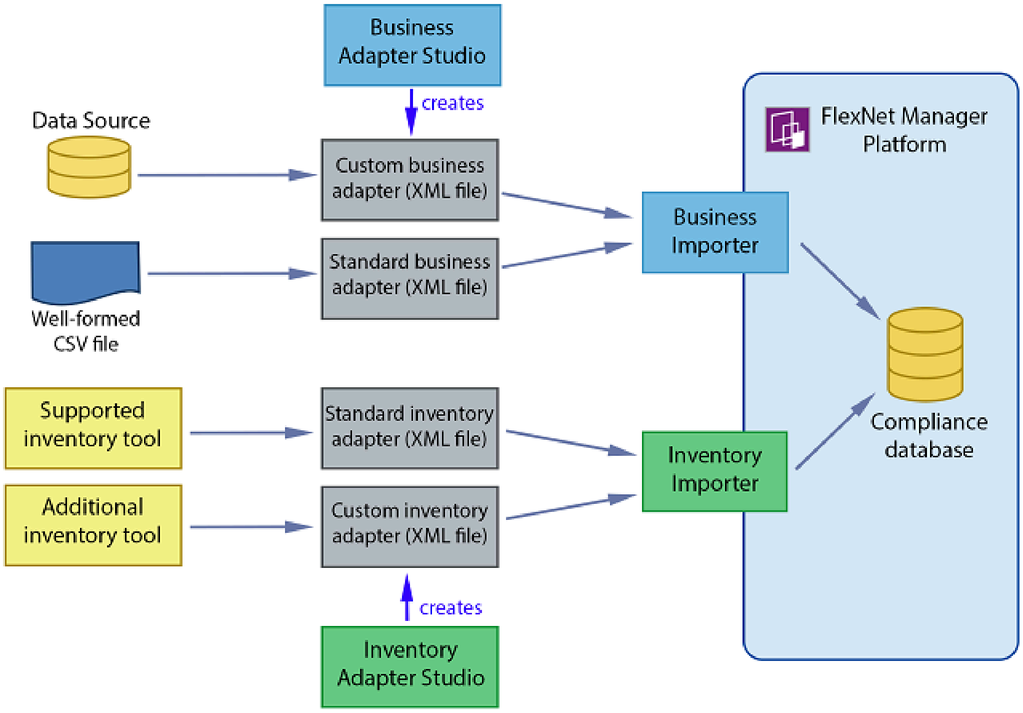
- **Installation:** The process is different for the two modes (and the two installation locations). The Business Adapter Studio for disconnected mode is installed with the inventory beacon. It is expected to run on the same computer as the inventory beacon, and makes use of other services that the beacon provides. By contrast, for connected mode, the Business Adapter Studio must be installed separately on your application server, and instructions are included in the installation documentation.

***Note:** If you self-install the Business Adapter Studio on your application server, you must upgrade your installation to the latest version to take advantage of new properties introduced in the latest data model.*

- **Skills:** Business Adapter Studio is intended for users comfortable with data models and mapping between them. It is an easy tool to use in that context, and provides guidance about available options. Templates are included that complete as much as possible of the definitions for you, and there are sample spreadsheets provided for those who prefer to standardize their datasets in that medium. You do not need SQL experience for disconnected mode adapters, as the Business Adapter Studio running on the inventory beacon does not allow for any custom SQL. In contrast, for connected mode, the Business Adapter Studio allows you to insert custom XML, but you should tackle this level of customization only if you are very confident that you will not cause disruption to your operational data (always use a test environment first!).

What is not suitable for a business adapter?

Business data does not include any inventory of hardware or software from your computer fleet. Inventory is imported during the inventory import process, for which a number of popular inventory tools are supported in a default implementation. You can also create inventory adapters to link non-standard inventory tools to the inventory import. You create inventory adapters using the separately available Inventory Adapter Studio, which is a separate tool from the Business Adapter Studio.



2

Overview: Development Process for Business Adapter

Business adapters import non-inventory data (such as purchases or enterprise structure) that helps to determine your compliance position. There are two separate modes in which business adapters can operate:

- In connected mode, where they run on your central application server, and therefore have direct access to your central operations databases.
- In disconnected mode, where they cannot connect to the operations databases because they run on an *inventory beacon* that is remote from it. Here you have some restricted capabilities to increase security.

Important: *In connected mode, data is injected directly into your operations database. You cannot see any evidence of the import in the* ***Business Data*** *tab of the* ***System*** > ***Data Inputs*** *page on your compliance console (this tab shows results only for imports from your inventory beacons).*

Both these modes are outlined in the same list below. If an item does not specifically mention connected or disconnected modes, then it is applicable to both.

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. In connected mode (only), configure the adapter's connection to the FlexNet Manager Suite database. This is where imported data will be written by business adapters running in connected mode. See Connecting to the Compliance Database (Connected Mode Only).
3. Configure the connection to the data source. This is where data will be imported from. See Connecting to a Data Source.
4. Confirm that you are querying the correct data from the data source. See Reviewing Data from the Source.
5. Load the list of properties from the data source, so that they can be mapped to objects in FlexNet Manager Suite. See Retrieving the List of Fields.
6. Link the objects in your source data to the objects that you want to update in FlexNet Manager Suite. See Choosing Target Database Items in FlexNet Manager Suite.

7. Define rules that manage updates and creation of these objects, based on the incoming data. See Defining Import Rules for a Database Item.
8. Define a mapping between the properties in the data source to those of the objects in FlexNet Manager Suite. See Defining Import Rules for Attributes/Properties.
9. In connected mode, you can simulate the data import to confirm it is working as expected. If necessary, you may use logging and tracing to debug any issues with the business adapter. Access all of these topics through Testing and Diagnosis for Your Business Adapter.
10. Save the adapter (Saving Business Adapters).
11. In connected mode, perform a real import from the data source into FlexNet Manager Suite (this calls the Business Importer to do the import). See Running an Import from the Business Adapter Studio.
12. Outside Business Adapter Studio, for connected mode only, configure a scheduled task to regularly run your new adapter. See Setting Up Regular Imports (Connected Mode).

3

Managing Business Adapters

You can create, save, modify, and test adapters from within Business Adapter Studio. Access the Business Adapter Studio itself through your inventory beacon in disconnected mode, or through the Windows Start menu on your central application server.

While business adapters can be entirely managed within the Business Adapter Studio, they may also be deleted in Windows Explorer. If you choose this approach, remember to modify any associated scheduled tasks as well.

To Start the Business Adapter Studio

For adapters that run on an inventory beacon, start the Business Adapter Studio from your inventory beacon interface. Alternatively, for those that will run on your central application server, start it from the Windows Start menu. Disconnected mode is used whenever the business adapter runs on a separate computer that is not the operations server for an on-premises installation of FlexNet Manager Suite. A typical example is where the business adapter will run on an inventory beacon, perhaps because of network partitioning that separates the data gathering from your central server.

When the adapter runs on your central, on-premises server and has simultaneous access both to the business data and to the operations database for FlexNet Manager Suite, this is called connected mode. There are separate methods for starting Business Adapter Studio in connected and disconnected modes.

***Note:** The account running the FlexNet Beacon interface requires administrator privileges. In particular, when running the Business Adapter Studio, the account must have write privileges to the registry on the server where it is executing. If this privilege is not available, and you select the encryption option in the Business Adapter Studio, the product will fail with the error* `The type initializer for 'Flexera.BusinessImport.BusinessImporterCryptgrapher' threw an exception.`

To start the Business Adapter Studio:

1. In disconnected mode, where the finished adapter will run on an inventory beacon:

    a. Select the **Business Importer** tab in the user interface for the inventory beacon.

    b. Optionally, if you think that new templates and reference files may be available since you started the inventory beacon user interface (UI), you may click **Download Configuration**.

As the configuration files don't change often, and are checked for currency each time that you start the inventory beacon UI, this is necessary only in special circumstances.

c. Click one of the following buttons:

- Click **New...** if you are starting development of a new business adapter.
- Click **Edit...** to modify one of your existing business adapters.

Business Adapter Studio displays an initial dialog to collect details, and opens the appropriate business adapter in the editing environment.

2. In connected mode, where the business adapter will run on your on-premises operations server:

a. From the Windows Start menu, open the **Flexera Software** program group.

b. In that group, click **Business Adapter Studio**.

Business Adapter Studio opens a blank window.

Creating a New Adapter

From the inventory beacon UI, you can start working on exactly one new business adapter at a time. Once the Business Adapter Studio is open, however, you can start other new adapters, and work on each one in its own tab.

When the Business Adapter Studio is already open:

1. Do either of the following:

- Click the New icon ( ) in the tool bar.
- From the **File** menu, click **New...**.

A shell for a new adapter is created in its own tab within Business Adapter Studio, and given a default name in the structure outline on the left (you can rename the adapter at any time).

Starting instead from the inventory beacon user interface:

2. In the user interface for the inventory beacon, select the **Business Importer** tab.

3. Click **New...**.

Business Adapter Studio displays an initial dialog to collect details.

4. Select the appropriate **Adapter template** from the option list.

The adapter templates correspond to the objects in the operations databases that you are allowed to import in disconnected mode (through an inventory beacon). Each type allows you to import a fixed set of attributes for that object, and sometimes a small set of links to other related objects in the database.

5. Give the business adapter a useful name (**Adapter name**) that will assist you with future maintenance. The adapter name will also be referenced by the Business Importer.

6. Choose how the finished business adapter will execute on its business connection to the third-party system in **Execute as**:

- Choose **Windows (current account)** if the connection will use the account that is then executing the FlexNet Beacon engine
- Choose **Windows (specific account)** if the Business Importer should use a different account to make the connection to the other system, or file share, and so on.

 If you choose the latter, the **Username** and **Password** fields are enabled, where you can provide the credentials for the specific account.

7. Click **Save**.

 A shell for a new adapter is created in Business Adapter Studio, ready for you to identify the connection to be used to gather the information.

Editing an Existing Business Adapter

Different choices are available depending on whether the Business Adapter Studio is already open.

You may be reopening an adapter that you have been working on recently, or you may be updating a business adapter originally created with an earlier release of FlexNet Manager Suite. As certain objects and attributes may be deprecated over time, you may see various alerts.

Use this process when starting from the inventory beacon (when the Business Adapter Studio is not already open):

1. In the inventory beacon, select the **Business Importer** tab.
2. In the list of **Current scheduled imports**, select the row identifying the business adapter you want to edit.
3. Click **Edit...**, and the **Edit business connection** dialog appears. (For more information about completing this dialog, see Creating a New Adapter.)

Choose either of these options when the Business Adapter Studio is already open (in either mode, on the inventory beacon or the application server):

- Click the Open icon (📂) in the tool bar.
- From the **File** menu, click **Open...**.

If you open an old business adapter that contains deprecated content, a large warning dialog appears, and the status bar displays the list of deprecated objects or properties. Click on the status bar to display the list more conveniently. Deprecated objects and properties have a different icon in the tree list (on the left hand side), and also have a text explanation such as `(Deprecated property)` alongside the name.

Best practice is to check the adapter and delete deprecated objects and properties.

***Tip:** Business adapters containing deprecated objects and properties can still be executed, but the behavior may be unpredictable and you are at risk of a failed execution.*

Renaming a Business Adapter

You can name (or rename) a business adapter at any time during the development process.

The starting conditions, and the UI presentation, vary:

- In disconnected mode (when the business adapter operates on an inventory beacon), you gave the business adapter a name as you created it. This adapter name is shown as the top-most node in the structure tree on the left.
- In connected mode (when the business adapter is running on your operations server in an on-premises implementation), the business adapter received a default name as you created it. The adapter name is the second level of the structure tree. Consider renaming this as you start your editing process.

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

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

Restriction: *On an inventory beacon in disconnected mode, you must not change the folder in which business adapters are saved.*

To save the business adapter you are working on now, do either of the following:

- From the **File** menu, choose **Save**.
- From the tool bar, click the Save icon (💾).

To save all the business adapters currently open in Business Adapter Studio:

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

Tip: *In either of these processes when you are in connected mode, if a business adapter has not previously been saved and remains unnamed, you are offered the chance to name it while saving. The name you supply is shown in the tab for this business adapter, if you have multiple adapters open.*

To change the XML file name, or (in connected mode only) save to a new location:

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

Tip: *If you are on your central application server and relocating an operational business adapter, but intend to continue its use from the new location, don't forget to check any existing scheduled task that may reference its old name/location, and then remove the old file.*

Keep in mind that renaming the XML file is a separate thing from changing the operating name of the business adapter itself (for which, see Renaming a Business Adapter).

Closing Business Adapters (and the Business Adapter Studio)

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.

4

Defining Connections for a Business Adapter

Every business adapter needs a connection defined to the external repository of business data, from which information will be read. This is its connection to the *source*.

Beyond that, connection requirements are different in connected mode and disconnected mode.

- In connected mode (where the business adapter is running on the application server), the business adapter also needs a second connection, to the operations databases (specifically the compliance database) where imported data will be written. This is its connection to the *target* database.
- In disconnected mode (where the business adapter is running on the inventory beacon), the business adapter has no access to the operations databases. Instead, business information is archived into packages and uploaded to the application server, where it is processed nightly and finally loaded into the operations databases (specifically the compliance database). This does not require any connection details. (If you have scaled up to a multi-server implementation, the upload is to the processing server.)

Use the same processes both for creating new connections, and for modifying connection details when you edit an existing adapter.

Connecting to a Data Source

You can prepare adapters for a wide variety of external data sources, and the details required depend on the kind of source 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.

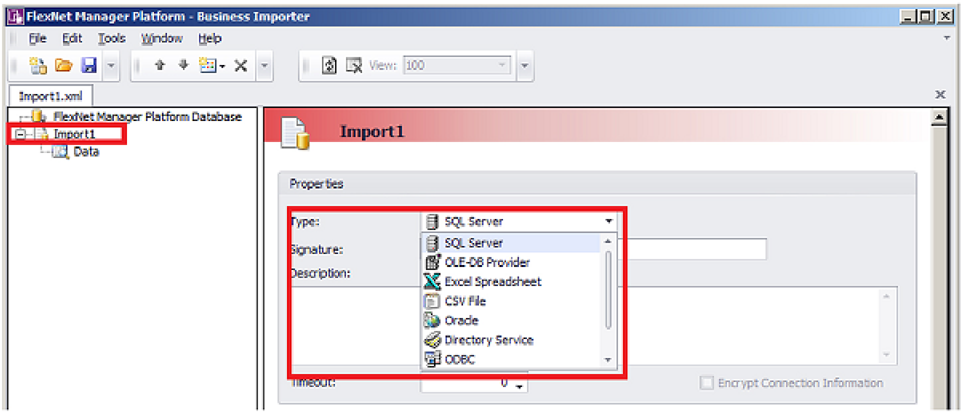
To initialize a data connection:

1. Ensure that the desired adapter is open in Business Adapter Studio.
2. In the structure tree on the left, select the node identifying your business adapter (in disconnected mode, this is the topmost node; in connected mode, it is at the second level).

A page of properties for your business adapter displays.

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 1: Choosing the **Type** for the data source



Type: SQL Server

Timeout: 0

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 computer where the business adapter runs:

- In disconnected mode, on the inventory beacon
- In connected mode, possibly on the application server.

These registry entries are found under the following registry key:

- For 32-bit operating systems:
  `[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Jet\4.0\Engines\Excel]`
- For 64-bit operating systems:
  `[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Jet\4.0\Engines\Excel]`

Entry	Notes
TypeGuessRow	The format for each column is automatically assigned, based on a sampling of 8 rows. This may generate issues in some scenarios. For instance, if the 8 first rows of specific column include only numeric values, the column will be considered as numeric when string values may exist in other rows below. Depending on the scenario, this may cause import errors or values may be discarded. One way to solve the problem is to change the number of rows considered by Excel. To modify this behavior, adjust the entry `[TypeGuessRow]`. This value defines the number of rows to read to determine the format of a column. A value of zero indicates the full Excel worksheet will be read; this value may impact performances.
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

Importing data into the FlexNet Manager Suite database from CSV (Comma-Separated Values) files (or plain 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:** 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

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 delimiter.

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:

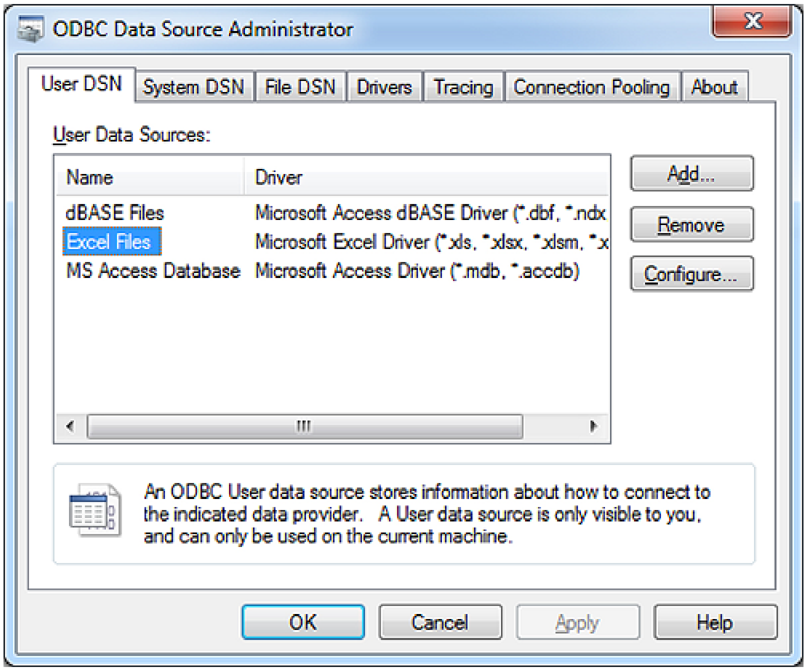
1. To initiate the process, do one of the following:
    - On a 32-bit machine, access the Windows Control Panel and select **ODBC** from the icons in the control

panel.

- On a 64-bit machine, run the following command: `C:\Windows\SysWOW64\Odbcad32.exe`.

The **ODBC Data Source Administrator** properties display.

Figure 2: The ODBC Data Source Administrator screen



Name	Driver
dBASE Files	Microsoft Access dBASE Driver (*.dbf, *.ndx
Excel Files	Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.x
MS Access Database	Microsoft Access Driver (*.mdb, *.accdb)

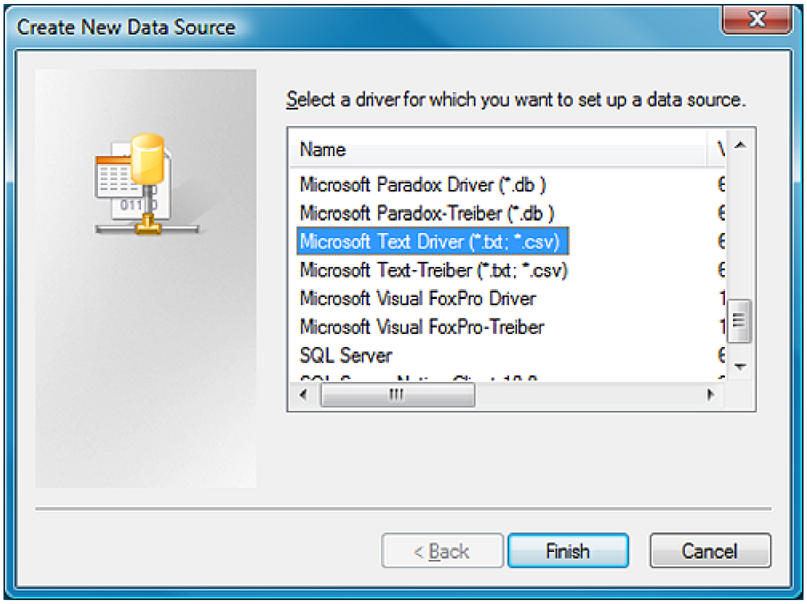
An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

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 displays.

Figure 3: Pick the driver matching your data source

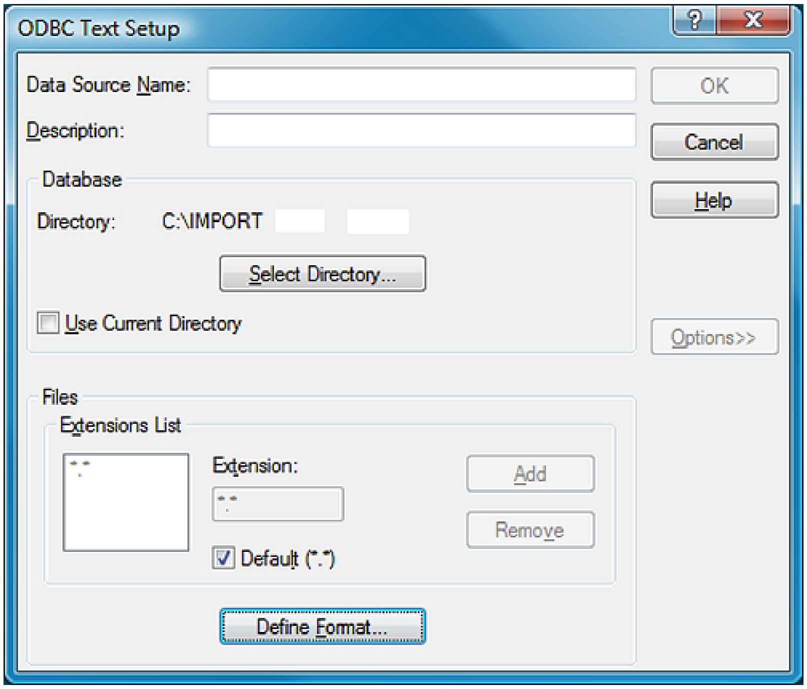

Create New Data Source
Select a driver for which you want to set up a data source.
Name
Microsoft Paradox Driver (*.db)
Microsoft Paradox-Treiber (*.db)
Microsoft Text Driver (*.txt; *.csv)
Microsoft Text-Treiber (*.txt; *.csv)
Microsoft Visual FoxPro Driver
Microsoft Visual FoxPro-Treiber
SQL Server
< Back
Finish
Cancel

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

4. Click **Finish**.

   The **ODBC Text Setup** dialog displays.

Figure 4: Locate the CSV (or text) file



Directory: C:\IMPORT

□ Use Current Directory

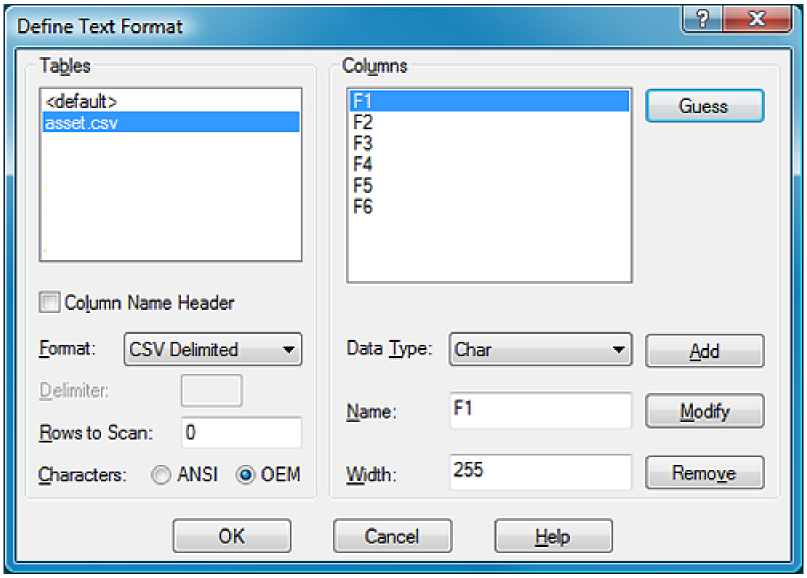
☑ Default (*.*)

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 displays.

Figure 5: Define the format of the text file



Tables: <default>, asset.csv

Columns: F1, F2, F3, F4, F5, F6

☐ Column Name Header

Format: CSV Delimited

Delimiter:

Rows to Scan: 0

Characters: ○ ANSI ● OEM

Data Type: Char

Name: F1

Width: 255

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.*
Password	The password (if required) for the account connecting to the directory service.

Property	Notes
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: `(&amp;(objectCategory=user)(objectClass=person)(mail=*))`
Referral 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

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.*

Connecting to the Compliance Database (Connected Mode Only)

When you run your own application server, you can build business adapters that connected directly to your compliance database.

The connection to the compliance database is most conveniently made when Business Adapter Studio is installed on the same server as FlexNet Manager Suite (in this case you do not need to know any connection details). When the installations are on separate computers, it is helpful if the development computer has network access to the compliance database. If not, you can work offline without an active connection to the compliance database, but the following limitations apply:

- You cannot load existing custom properties
- You cannot see values in lookup tables
- You cannot simulate data imports for testing, nor trigger real imports

- You cannot access the history of attempted data imports.

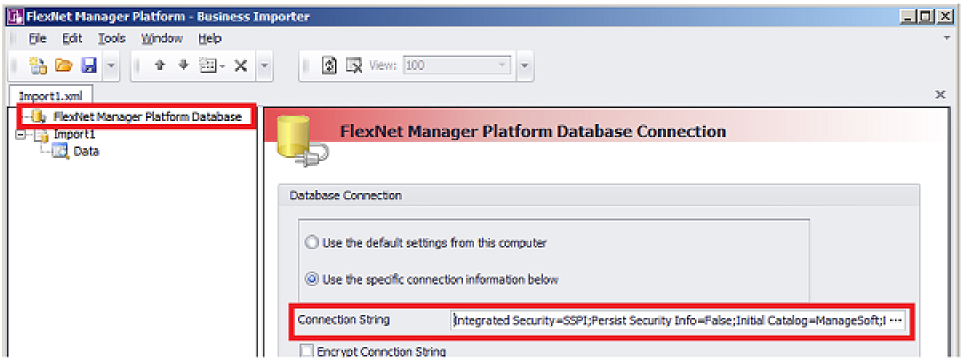
To connect to the compliance database (connected mode only):

1. Ensure that the desired business adapter is open in Business Adapter Studio.
2. In the structure tree on the left, select the top node (**FlexNet Manager Suite Database**).

   A page for connection details displays.
3. Select one of the following options:
   - **Use the default settings from this computer** — Select this option only when Business Adapter Studio is running on the same server as FlexNet Manager Suite. This option reads the connection details for the local compliance database from the registry key `SOFTWARE\ManageSoft Corp\ManageSoft\Reporter\CurrentVersion\DatabaseConnectionString`. If you have chosen this option, your definition is complete.
   - **Use the specific connection information below** — Select this option when Business Adapter Studio is running remotely from the compliance database (but still in connected mode), or when you are configuring an adapter for an instance of Business Importer that runs remotely from the compliance database (but still in connected mode). This option requires you to complete the remaining steps.

   Important: *Do NOT attempt to use this setting for business adapters that run on inventory beacons in disconnected mode.*

   Figure 6: Defining the connection string to the compliance database

   FlexNet Manager Platform Database Connection

   Database Connection

   ○ Use the default settings from this computer

   ● Use the specific connection information below

   Connection String: Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=ManageSoft;I ···

   □ Encrypt Connction String

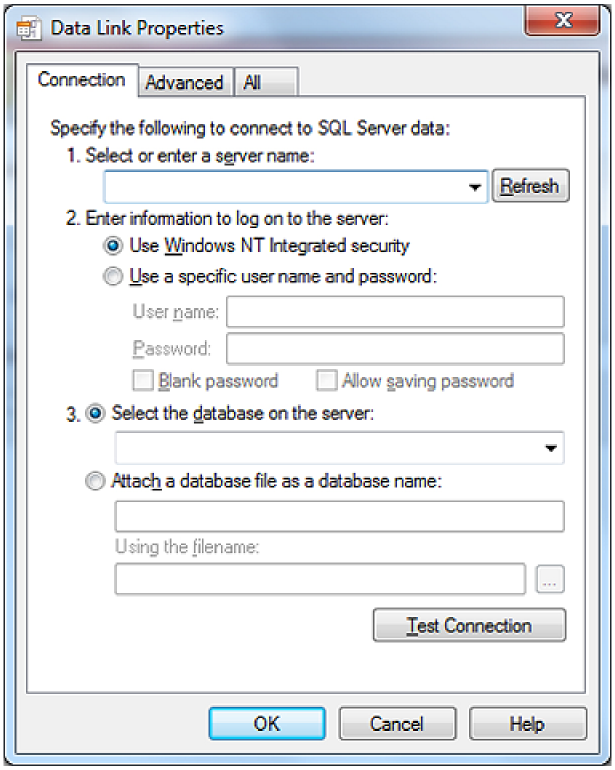
   FlexNet Manager Platform - Business Importer
File
Edit
Tools
Window
Help
View: 100
Import1.xml
FlexNet Manager Platform Database
Import1
Data

4. To complete the **Connection String** field, click the ellipsis button (**...**) at the right end of the field.

   The standard Microsoft Windows **Data Link Properties** dialog appears.

   Tip: *Although using the **Data Link Properties** dialog is strongly recommended, it is possible to manually edit the connection string when required.*

Figure 7: Specify connection to the compliance database


Data Link Properties
Connection
Advanced
All
Specify the following to connect to SQL Server data:
1. Select or enter a server name:
Refresh
2. Enter information to log on to the server:
● Use Windows NT Integrated security
○ Use a specific user name and password:
User name:
Password:
□ Blank password
□ Allow saving password
3. ● Select the database on the server:
○ Attach a database file as a database name:
Using the filename:
...
Test Connection
OK
Cancel
Help

5. Complete the details in the **Data Link Properties** dialog:

   a. In **Select or enter a server name**, choose or enter a fully qualified server name (or IP address) for the server on which the database is running.

   b. Choose the authentication method for the account under which the Business Importer will access the database. **Use Windows NT Integrated Security** is recommended for easier maintenance over the long term; or you may use SQL authentication by choosing **Use a specific user name and password**, and entering the account details.

   > ***Important:*** *If you enter the account details, be certain to select* ***Allow saving password*** *so that the password can be brought into the adapter. Otherwise the password will be lost as soon as you click* ***OK****.*

   c. From the **Select a database on the server** 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.

If you selected Windows authentication, a typical connection string is of the form `Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=ManageSoft;Data Source=`*`SQLServerName`*

For SQL authentication, a typical connection string is of the form `Password=`*`SQLPassword`*`;Persist Security Info=True;User ID=`*`SQLAccount`*`;Initial Catalog=ManageSoft;Data Source=`*`SQLServerName`*

__Tip:__ A period (.) entered as the `Data Source` in the connection string uses the database connection on the current server where the Business Importer is running.

6. Select the **Encrypt Connection String** check box to encrypt the connection string details stored in the XML file for this adapter.

7. Save the adapter.

5

Reviewing Data from the Source

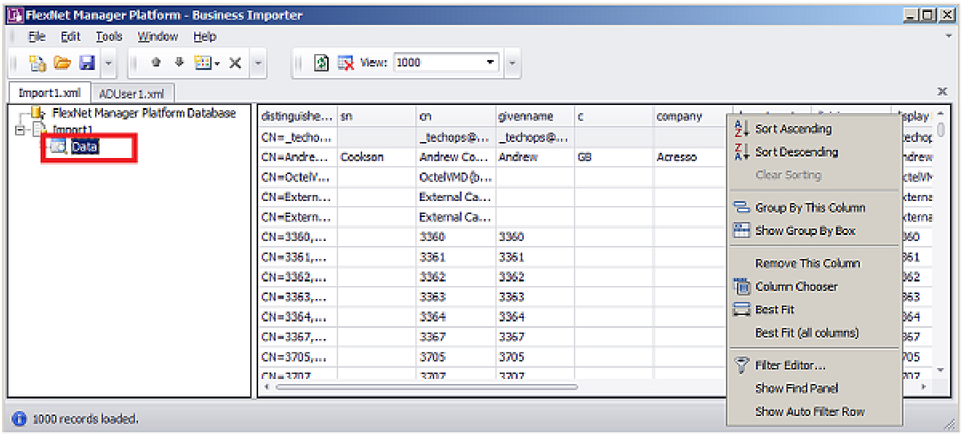
Once the data source connection has been configured, the next step is to preview the data returned from the query. This confirms that you have configured the connection properly and are retrieving the expected data. You can validate the returned data, with no effect on the source data or future imports.

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 8: The data grid offers many options for organizing the list



distinguishe...	sn	cn	givenname	c	company	splay
CN=_techo...		_techops@...	_techops@...			echop
CN=Andre...	Cookson	Andrew Co...	Andrew	GB	Acresso	ndrew
CN=OctelV...		OctelVMD (b...				ctelVM
CN=Extern...		External Ca...				xterna
CN=Extern...		External Ca...				xterna
CN=3360,...		3360	3360			360
CN=3361,...		3361	3361			361
CN=3362,...		3362	3362			362
CN=3363,...		3363	3363			363
CN=3364,...		3364	3364			364
CN=3367,...		3367	3367			367
CN=3705,...		3705	3705			705
CN=3707		3707	3707			707

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.*

6

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.

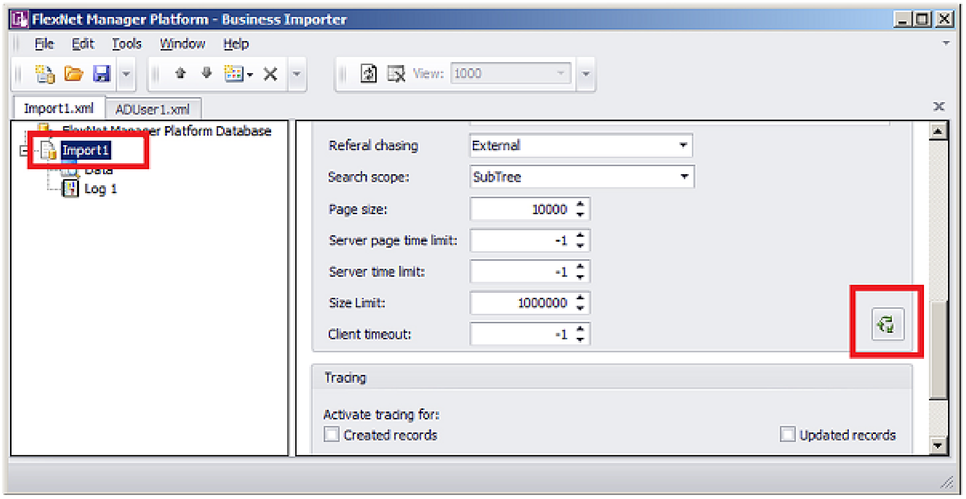
Retrieving the List of Fields

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 9: The retrieval button fetches the field list into memory



Referal chasing: External

Search scope: SubTree

Page size: 10000

Server page time limit: -1

Server time limit: -1

Size Limit: 1000000

Client timeout: -1

Activate trading for:

□ Created records □ Updated records

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:*

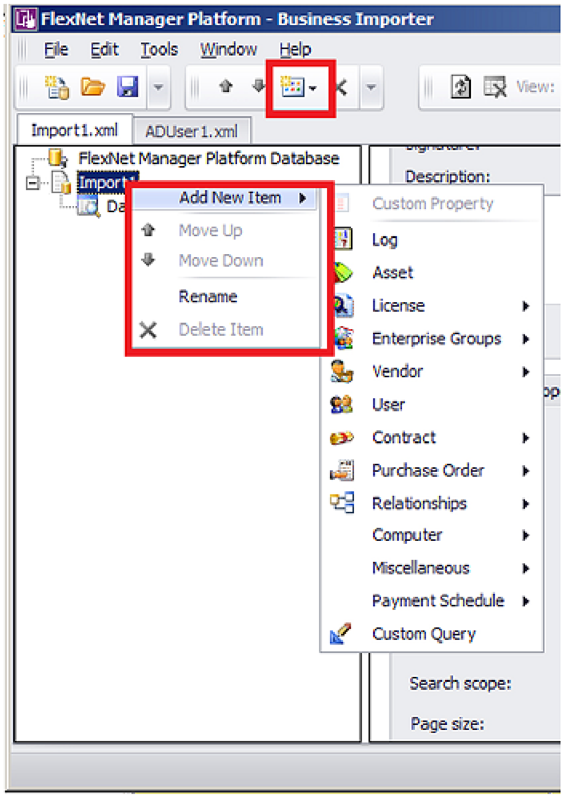
- *The custom property has been defined on the application server*
- *You have downloaded the latest templates and data schema that includes your custom properties. For details, see Updating Business Adapter Templates and Data Model.*

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 10: 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

Ensure your business adapter is open in Business Adapter Studio.

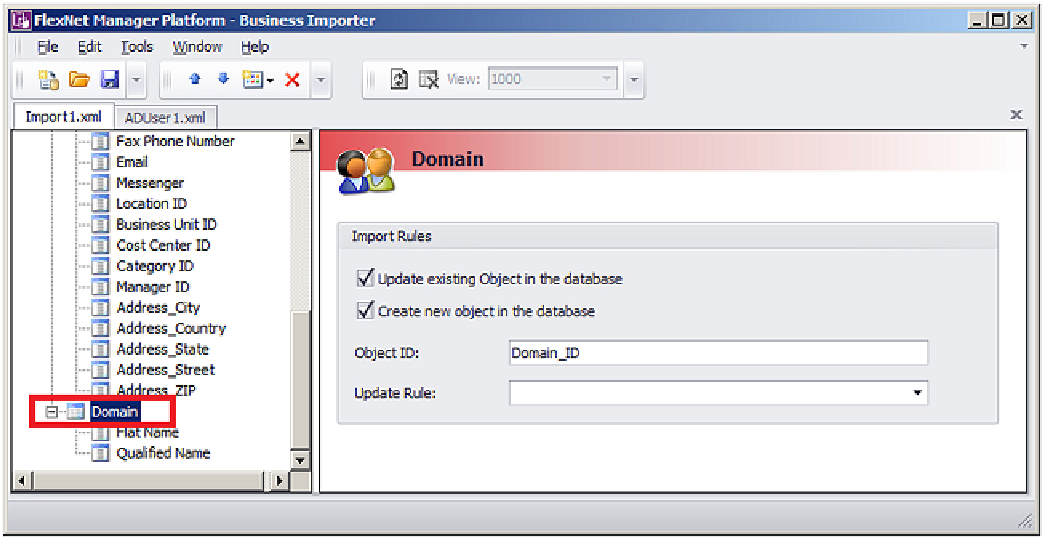
Import rules may be set separately for objects and attributes. This topic covers the higher-level objects/items.

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 11: The import rules for an object in the compliance database

   ☑ Update existing Object in the database

   ☑ Create new object in the database

   Object ID: Domain_ID

   Update Rule:

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).
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 ◦ `Detach all unfound links between the considered licenses and PO`

Option	Description
	`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 ◦ `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

Option	Description
	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

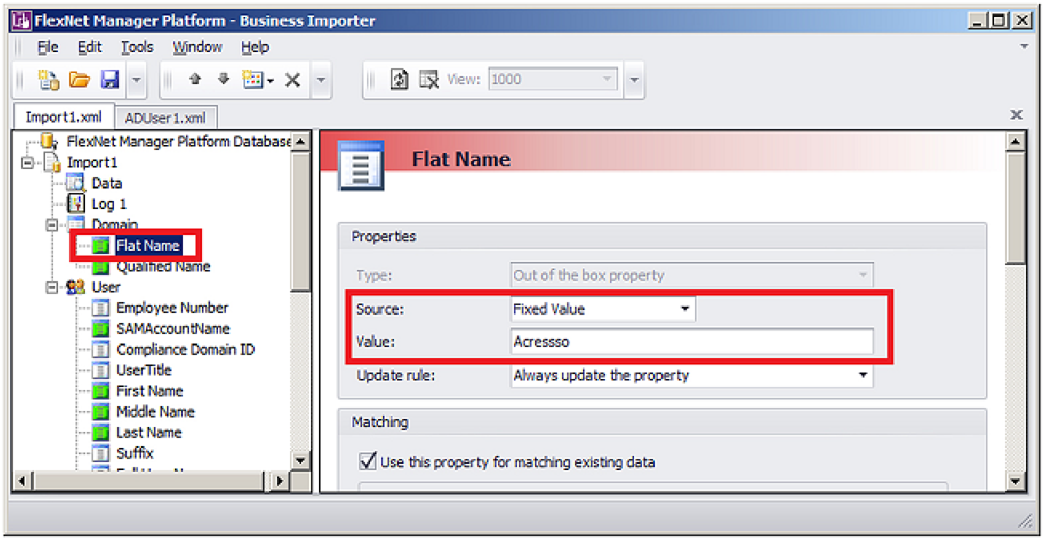
When you have set the import rules for an item in the compliance database/>, you may set fine-grain import rules for handling imports to the individual properties of each of those objects.

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 12: Import rule for a *property* of compliance database/> object



Type: Out of the box property

Source: Fixed Value

Value: Acressso

Update rule: Always update the property

☑ Use this property for matching existing data

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 FlexNet Manager Suite web interface
- `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.
- `SQL Value` — The result of an SQL query. The **Value** field must contain a SQL fragment which will evaluate to a value. The SQL fragment may include columns from the data source.

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

In connected mode, this setting works in conjunction with **Matching mode** and **Value pattern**. It is relevant only when **Matching mode** is `Like` (and is otherwise ignored). For `Like` matches on key field data, this setting is a pattern to be matched in the *existing target compliance database/>*.

The rules for specifying a pattern match are:

- The wild card character is the percent sign (%).
- A literal value must be enclosed in square brackets, as `[value]`.
- The way of combining the wild card and literal value depends on the setting for **Source** (in the **Properties** group, above). For example, to create a `Like` match that expressed the concept of 'contains *value*', write the pattern in the following ways:
    - If **Source** is `Fixed Value`, write: `%[value]%`
    - If **Source** is `SQL Value` or `Field Value`, write: `'%' + [value] + '%'`

In disconnected mode, on your inventory beacon, this setting is disabled, and is always interpreted as `[value]%`.

Matching mode

The type of matching to perform. This setting works in conjunction with **Property pattern** and **Value pattern**. Possible values are:

- `Equal` — The value of the existing compliance database/> property must exactly match the incoming value in the imported data
- `Like` — Enables matching with the use of wild cards on either side of the test, with the compliance database/> side expressed in **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: `(?&lt;=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. • Enter an SQL statement to customize the computer matching behavior. 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 `BusinessImportLogDetail` table processing the import • `[ImportObjectID]` — The ID of the record in the `BusinessImportLogObject` 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 `BusinessImportLogObject` table.

6. As you make your changes on this page, your specification is saved in memory. When you are ready, click the Save button in the tool bar, or choose one of the saving options from the **File** menu.

Creating Custom SQL

At any stage in your adapter, which runs from top to bottom down the structure tree, you can insert some custom SQL to modify (or make log entries for) the data that has (so far) been written into the compliance database in FlexNet Manager Suite. You may include:

- SQL statements

- Stored procedures.

Any custom SQL is run as part of the same transaction as the import of the external data.

To create custom SQL (connected mode only):

1. Right-click your adapter's name in the structure tree.
2. In the context menu, select **Add New Item**, and from the submenu select **Custom Query**.
3. To give your query a meaningful name (making it easier to read and maintain in the finished adapter XML file):
    a. Right click the new **Custom Query** entry in the structure tree
    b. Select **Rename** from the context menu
    c. Provide a new name.
4. Enter your SQL statement into the **Query Text** field.

    Tip: *You may include the keyword `[LOG_IMPORT_ID]` to return the current value of the `ImportID` column in the `BusinessImportLogSummary` table.*

    For example:

    - To run a stored procedure called *`MyStoredProcedure`* (optionally with parameters), enter

    ```
    EXEC MyStoredProcedure Parameter1,...
    ```

    - To post an entry in the log for a custom object (which will not otherwise be logged), enter

    ```
    Insert into BusinessImportLogObject([ImportID], [ObjectName], [StartDate],
                                [EndDate], [ObjectType], [Status]) values
    ([LOG_IMPORT_ID],
                                'My Custom Object', getdate(),
                                getdate(),'Custom', 1)
    ```

5. In the **Timeout** field, set the number of seconds that the Business Importer should wait before giving up waiting for a response to the custom SQL statement. The following values have special meaning:
    - `-1` means use the default time out for SQL Server
    - `0` means there is no limit and the Business Importer will wait indefinitely for the query to finish.
6. If you wish to write a custom message in the log file as part of this custom SQL, select the **Send the internal log ID as a parameter** check box. When checked, the `@FinImporterLogID` parameter value is automatically injected to the stored procedure as it is executed. The value of this parameter is the current record ID from the `FinImporterLog` table (automatically created).

Adding Custom Properties

Business adapters that run in connected mode (that is, on your central application server) can create custom properties in your compliance database on the fly.

FlexNet Manager Suite allows the addition of custom properties to existing objects in the compliance database. You can do this using the web interface. However, since it is very likely that the need for custom properties becomes apparent when integrating external data sources, you can also specify custom properties as a part of building a business adapter, as long as this adapter runs in connected mode on your operations server. The first time that the Business Importer runs this business adapter, the new custom property is created. On subsequent imports, data can be added to the custom property in the same way as for all other properties (the adapter does not have to be modified to remove the creation of the custom property).

Tip: *Remember that you must create the custom property before you can populate it with data — meaning that the database object with the custom property attached must be high in the structure tree.*

Note: *Any business adapter that creates custom properties can be run only on the application server. Once custom properties are created (through either means), you can also populate those properties with values through another business adapter running in disconnected mode on the inventory beacon.*

To add a custom property:

1. In the structure tree on the left of the Business Adapter Studio, right-click the compliance database object for which you want to create a custom property.
2. From the context menu, select **Add New Item**.
3. From the submenu, select **Custom Property**.
4. To give your custom property the name that will be created in the FlexNet Manager Suite compliance database:
   a. Right-click the new entry **Custom Property *n*** in the structure view.
   b. Select **Rename**.
   c. Type the new name, and press `Return`.

The details in the main data page are the same for this custom property as for all other properties (see Defining Import Rules for Attributes/Properties).

Triggering Immediate Update of the BAS Data Model

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

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

To update the data model now:

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

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

   The default value is

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

3. Execute the following command:

   ```
   BatchProcessTaskConsole.exe run BusinessAdapterConfig
   ```

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

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

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

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

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

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

   c. Edit an existing import, or create a new one, and click **Edit adapter...**.

      Your newly-created custom properties are included in the list of available properties with a distinctive icon. ▤

7

Testing and Diagnosis for Your Business Adapter

In connected mode, before running any import (even to a test database, let alone a production environment), you can run a simulation that will show the projected results of the Business Importer running your new adapter.

In connected mode, you will also want to create log files for your imports, and possibly run tracing. All these topics are included in this section.

Testing and diagnosis options are limited for business adapters running in disconnected mode on your inventory beacon. After running the adapter against the source database, you can inspect the archive package that will be uploaded to the central application server. This package is saved in `Program Data\Flexera Software\Beacon\IntermediateData\`, and is a zip archive containing the following:

- A file DDI.xml that represents the business adapter (without its connection strings), so that you can see the steps than run in your adapter
- A manifest that includes a result code from running your adapter, and any error messages
- An XML file of the collected data.

On the inventory beacon, you may also examine the log file for the beacon engine, which includes results of uploading the intermediate package. This is saved in `%ProgramData%\Flexera Software\Compliance\BeaconEngine`. You may search in the log file for the name of your business adapter to find steps relating to it.

Note: *There is no specific presentation in the compliance console (the web interface) of the results of importing your business adapter data. Look for collected information displayed in appropriate lists after the import is completed. Of course, the impact on your compliance position that results from this imported business information is shown only after the next inventory import and reconciliation.*

Specifying a Log File

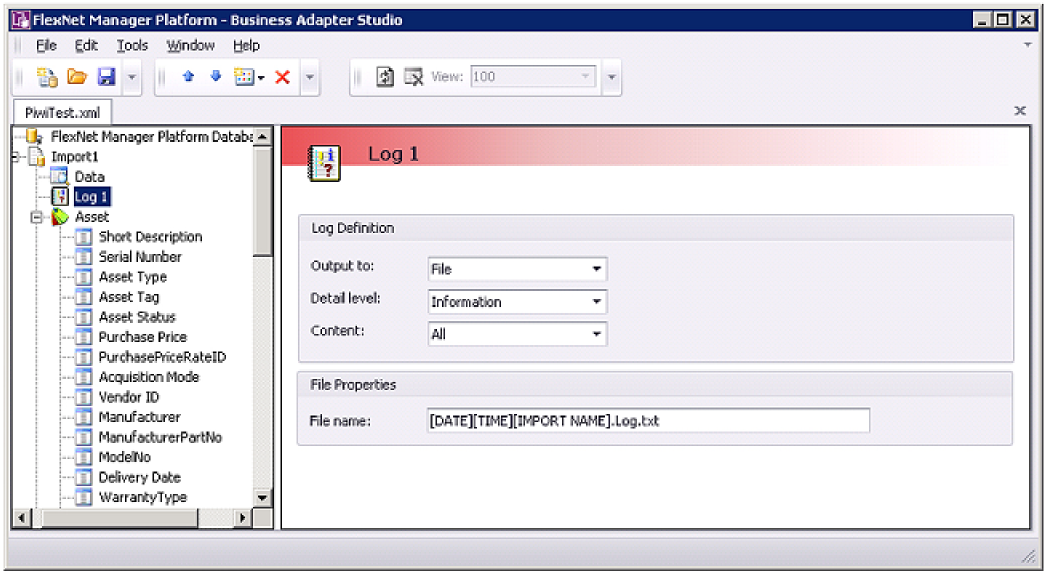
Log files help you monitor the operations of your business adapter. You can specify a log file that is used for all imports using this business adapter. The log that you specify is updated automatically by the Business Importer, overwriting the previous log file on each occasion.

To specify the log file details:

1. Right-click on the adapter name in the structure tree.
2. From the context menu, select **Add New Item**, and from the submenu select **Log**.

   A page of log file settings appears, populated with default values.

   Figure 14: The default values for a log file



Log Definition

Output to: File

Detail level: Information

Content: All

File Properties

File name: [DATE][TIME][IMPORT NAME].Log.txt

3. Adjust the following **Log Definition** settings to your requirements.

Setting	Notes
Output to	The type of log output to generate. Possible values are: • Console — Output appears on the console of the test computer during development of the business adapter. It is likely to be less useful when the production version of the business adapter is running on the inventory beacon or the application server. • File — Output will be saved to a text-based log file. When you select this option, additional fields appear at the bottom of the page to specify the file details. • Database — In connected mode only (not available on the inventory beacon), the log is written into the compliance database, in the following tables (available only in Microsoft SQL Server Management Studio, and not in the user interface for FlexNet Manager Suite): ◦ BusinessImportLogDetail — A record is written here for every database record that has tracing turned on (see Tracing) when the record is rejected, created, updated or deleted. ◦ BusinessImportLogObject — A record is written here for each compliance database standard object that is created or updated during an import. Custom objects are not logged automatically, although you can cause a log entry to be added for a custom object (see Creating Custom SQL). ◦ BusinessImportLogSummary — A record is written here each time the Business Importer is started, either for a data import or for a simulation during testing. The record includes a Status result of 0 for a task not completed (error), and 1 for success. • Tcp/IP socket — Output is written to a TCP/IP socket. When you select this option, additional fields appear at the bottom of the page to specify the TCP/IP details.
Detail level	The depth of logging to provide. Choosing a given level of logging includes information for all higher levels in this ordered list: • Silent — (highest value) No details are recorded for individual database objects, although a record of the overall import/simulation still appears in the summary table (for database logging). • Critical • Errors • Warnings • Information • Debug — (lowest value) The most detailed logging information is written.

Setting	Notes
Content	Master control for what is written to the log. The settings are: • `Detail` — Writes information for each object to the log according to the level specified in the **Detail level** setting (above) • `Summary` — At the end of the import, writes a summary for each affected object. • `All` — Combination of the `Summary` and `Detail` levels.

4. If you selected **Output to** `File`, specify the name of the log file in the **File name** setting.

   - If you clear this field, the default file name is `MBI.log.txt`.
   - You may include UNC path definitions, or relative paths starting from the directory where the Business Importer executable is running.
   - When no file path is specified, the default is a subfolder Log, below the Business Importer folder.

   Remember that a log file with the same name is overwritten at each import. You may wish to use the following placeholder variables to create a unique file name for each pass at least during test and development (log files with unique names are not automatically cleaned up, and you assume that responsibility):

   - `[DATE]` — the date the import was started (formatted as *yyyymmdd*)
   - `[TIME]` — the time the import was started (formatted as *hhmmss*)
   - `[IMPORT NAME]` — the name of the adapter used for this import, taken from the name set for the adapter in the structure tree of the Business Adapter Studio.

5. If you selected **Output to** `Tcp/IP Socket`, specify the settings in the **TCP/IP Socket Properties** section that appears:

   - **Server** — The server name where the listener is installed. For the same computer on which the Business Importer is running, use `localhost`.
   - **Port number** — The port where the listener is waiting.

Tracing

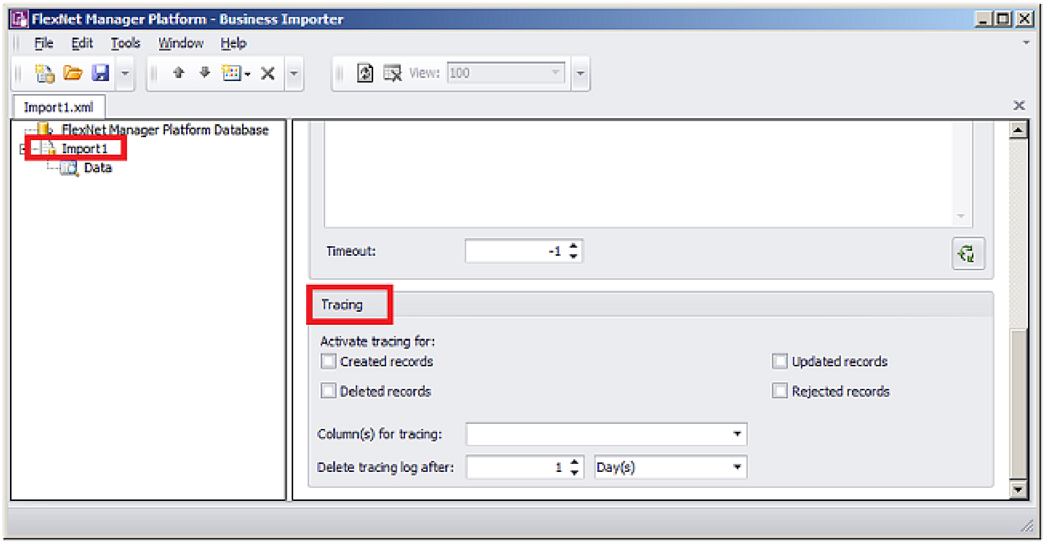
Tracing is available only for those business adapters running in connected mode (that is, on your central application server).

Tracing adds another stream of information, additional to logging outputs, to make debugging easier. It is controlled from the adapter node in the structure tree. Tracing outputs are written to the `BusinessImportLogDetail` table in the compliance database under FlexNet Manager Suite, regardless of the logging output type that you selected. You can inspect the rows of this table in Microsoft SQL Server Management Studio.

To configure tracing (connected mode only):

1. Select the adapter node in the structure tree, and scroll down the settings page to see the **Tracing** group.

Figure 15: Tracing settings are at the bottom of the settings page for the adapter



Timeout: -1

Delete tracing log after: 1 Day(s)

2. Adjust the following settings for tracing:

Setting	Notes
Activate tracing for	Select as many of the following check boxes as needed for the tracing you require for debugging your adapter. Note that if no check boxes in this group are selected, no tracing records are created. You may create a tracing entry each time that any record in the compliance database is created, updated, or deleted; or when a record in the source data is rejected. **Rejected records** are incoming data assessed for import but not accepted. A common reason for rejection is that a column (such as a serial number) has been specified for database matching, but that column is blank for a particular record.
Column(s) for tracing	By default, tracing identifies affected records only by their count number within the flow of incoming data from the external source. As this is not a convenient aid to debugging, you may also pick one or more columns from the incoming data to help identify the tracing records. For example, if you selected a column called `Serial No` from this drop-down list of the available columns in the data source, and the computer with serial number 19283645 was rejected, the tracing would identify `Record 238, Serial No 19283645` This is more convenient for identifying the offending record. Best practice is to use an identifying column that contains unique values and is always populated. You may either: • Select a single column name (in the external data source) from this drop-down list • Directly enter multiple column names in the field, enclosing each in square brackets and separating the bracketed values with a comma.

Simulating an Import of a Business

Adapter

When you run a simulation, Business Adapter Studio processes all of the data as if an import were taking place, and lists the number of records that would have been processed and the number of records that would be rejected.

This means that you can review the results of the simulated import, and, if necessary, refine the connection settings before doing an actual import and changing the FlexNet Manager Suite data.

A simulation does not load data into the object records in the compliance database in FlexNet Manager Suite, but it does update the history tables for every object that would have been affected were the import actually performed, to record that a simulation has taken place. These history records are visible in the **History** tabs of each object's property sheet.

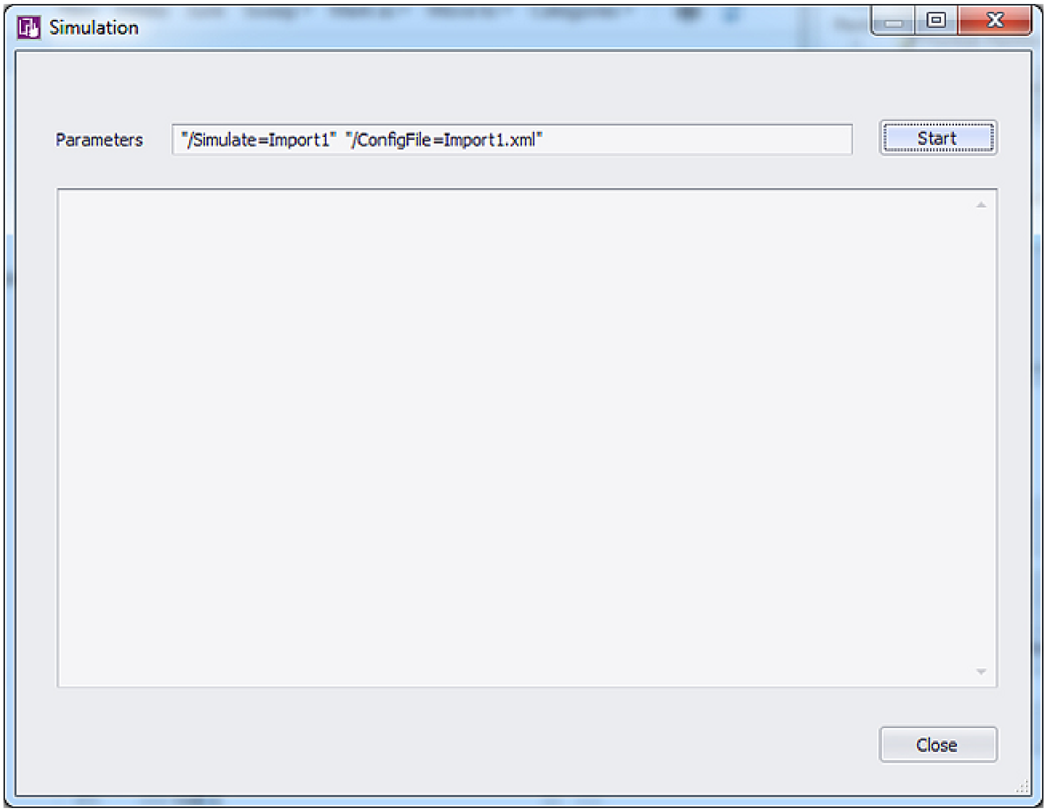
Once you are satisfied with the results of a simulated import, you can perform a real import. Even after performing a simulation, it is best practice to run the first import on a test database rather than in your production environment.

To simulate a data import:

1. Ensure you have selected the tab for the adapter you wish to test.
2. Select the **Tools > Simulate** menu options.

   The **Simulate** page displays. Do not change the default parameters.

   Figure 16: Run a simulation to check that the import file can be processed without affecting the FlexNet Manager Suite data

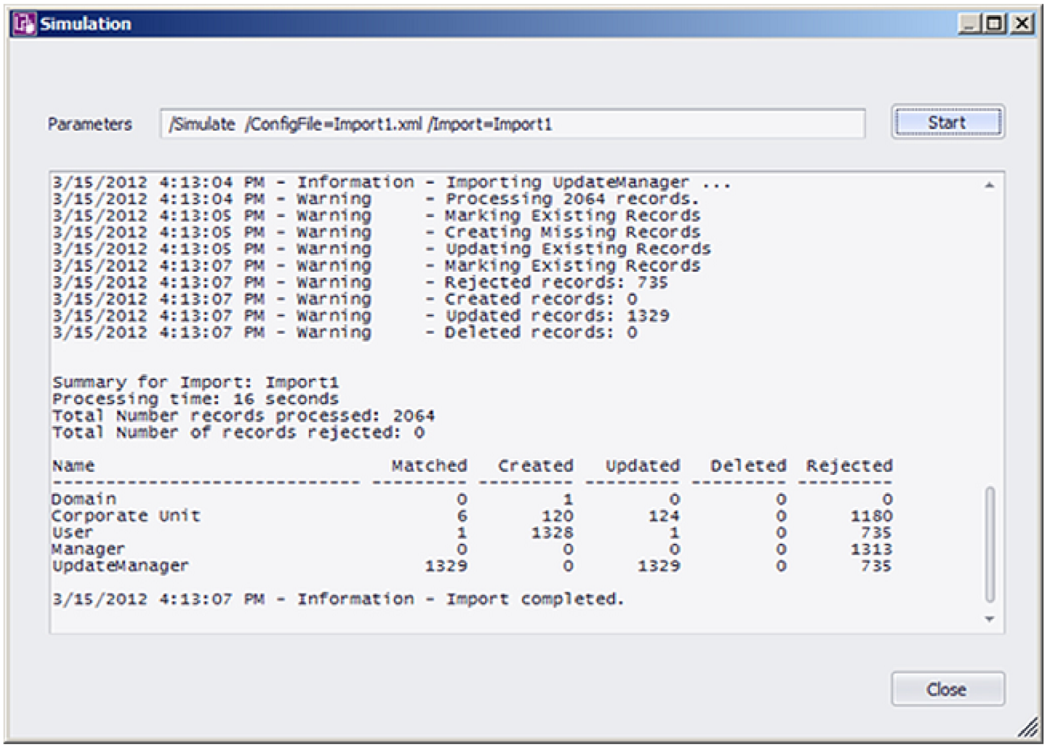


Parameters "/Simulate=Import1" "/ConfigFile=Import1.xml"

3. Click **Start**.

The simulated import is processed, and progress messages display on the window. When the import is complete, a summary of the records that could be processed in the simulation displays.

Figure 17: Review the results of the simulated process



Parameters /Simulate /ConfigFile=Import1.xml /Import=Import1

```
3/15/2012 4:13:04 PM - Information - Importing UpdateManager ...
3/15/2012 4:13:04 PM - Warning     - Processing 2064 records.
3/15/2012 4:13:05 PM - Warning     - Marking Existing Records
3/15/2012 4:13:05 PM - Warning     - Creating Missing Records
3/15/2012 4:13:05 PM - Warning     - Updating Existing Records
3/15/2012 4:13:07 PM - Warning     - Marking Existing Records
3/15/2012 4:13:07 PM - Warning     - Rejected records: 735
3/15/2012 4:13:07 PM - Warning     - Created records: 0
3/15/2012 4:13:07 PM - Warning     - Updated records: 1329
3/15/2012 4:13:07 PM - Warning     - Deleted records: 0


Summary for Import: Import1
Processing time: 16 seconds
Total Number records processed: 2064
Total Number of records rejected: 0

Name                             Matched   Created   Updated   Deleted  Rejected
------------------------------ --------- --------- --------- --------- ---------
Domain                                 0         1         0         0         0
Corporate Unit                         6       120       124         0      1180
User                                   1      1328         1         0       735
Manager                                0         0         0         0      1313
UpdateManager                       1329         0      1329         0       735

3/15/2012 4:13:07 PM - Information - Import completed.
```

4. Review the results. Check for any records rejected, and whether the correct number of records were processed as expected.

> ***Tip:*** *The most common reason for an incoming record to be rejected is that it lacks a value in a field that has been nominated for matching with existing records in the compliance database.*

5. When you have finished reviewing the results, click **Close**.

Comparing Result of Simulated Imports

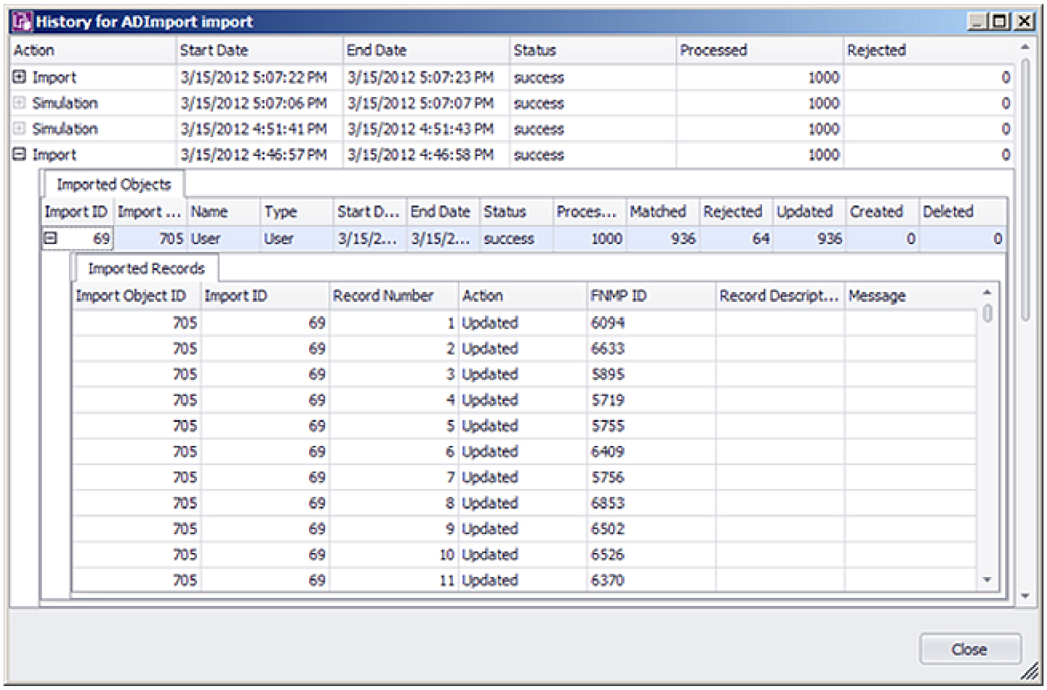
Once you have run several simulations, you may want to compare the overall results. A history of simulations is available, combined with the history of data imports.

To compare import results (simulations only):

1. With the tab selected for the correct adapter, select the **Tools > View History** menu options.

The **History for ...** page displays.

Figure 18: A list of past imports and simulations displays



Action	Start Date	End Date	Status	Processed	Rejected
⊞ Import	3/15/2012 5:07:22 PM	3/15/2012 5:07:23 PM	success	1000	0
⊞ Simulation	3/15/2012 5:07:06 PM	3/15/2012 5:07:07 PM	success	1000	0
⊞ Simulation	3/15/2012 4:51:41 PM	3/15/2012 4:51:43 PM	success	1000	0
⊟ Import	3/15/2012 4:46:57 PM	3/15/2012 4:46:58 PM	success	1000	0

Import ID	Import ...	Name	Type	Start D...	End Date	Status	Proces...	Matched	Rejected	Updated	Created	Deleted
⊟ 69	705	User	User	3/15/2...	3/15/2...	success	1000	936	64	936	0	0

Import Object ID	Import ID	Record Number	Action	FNMP ID	Record Descript...	Message
705	69	1	Updated	6094		
705	69	2	Updated	6633		
705	69	3	Updated	5895		
705	69	4	Updated	5719		
705	69	5	Updated	5755		
705	69	6	Updated	6409		
705	69	7	Updated	5756		
705	69	8	Updated	6853		
705	69	9	Updated	6502		
705	69	10	Updated	6526		
705	69	11	Updated	6370		

2. To view details of the records imported (or simulated) during one of the actions listed on the page, expand the icon to the left of the import action.

   Details of each record imported or updated (or the simulation of that action) display.

3. When you have finished reviewing the history, click **Close** to close the window.

4. If the results are as expected, your connection is ready for you to perform a real import. If the results are not as expected, you may need to modify the configuration for this adapter. See Linking Data Imports to FlexNet Manager Suite for more guidance.

Troubleshooting Business Adapters

Here are some possible issues and causes. Please advise any other cases that should appear here in future.

Issue	Notes
Imported data does not appear in custom views	There may be dependencies between different data fields. For example, if you are importing details about purchase orders than include prices, each price must have a corresponding rate (currency) identified. Without this correspondence, the numeric values are blanked out of the custom view. (For example, in the product schema, see `UnitPrice` and `UnitPriceRateID` in the `PurchaseOrderDetail` table.)
Monetary values appear without any currency units	The rate ID for this value is missing from your imported data.

8

Importing Data With Your Business Adapter

You have tested and debugged your business adapter. You have run a simulation and checked the imported values are as expected. Now you are ready to import data into your operations databases.

You may:

- Perform a single import for an adapter running in connected mode, where that import is controlled by the Business Adapter Studio
- Examine the history of imports you have run from the Business Adapter Studio, in connected mode
- Schedule regular imports for operations, which do not require the Business Adapter Studio
- Trigger a special, additional import from within the web interface, independent of the Business Adapter Studio.

Running an Import from the Business Adapter Studio

When you have simulated your import and you are satisfied with the results, you can perform a real import.

***Important:** When you import data, you make live changes to the FlexNet Manager Suite database. Incorrect settings may result in important data being deleted or modified. You should always test your import in a test environment or pre-production environment before importing into the live data.*

Important:

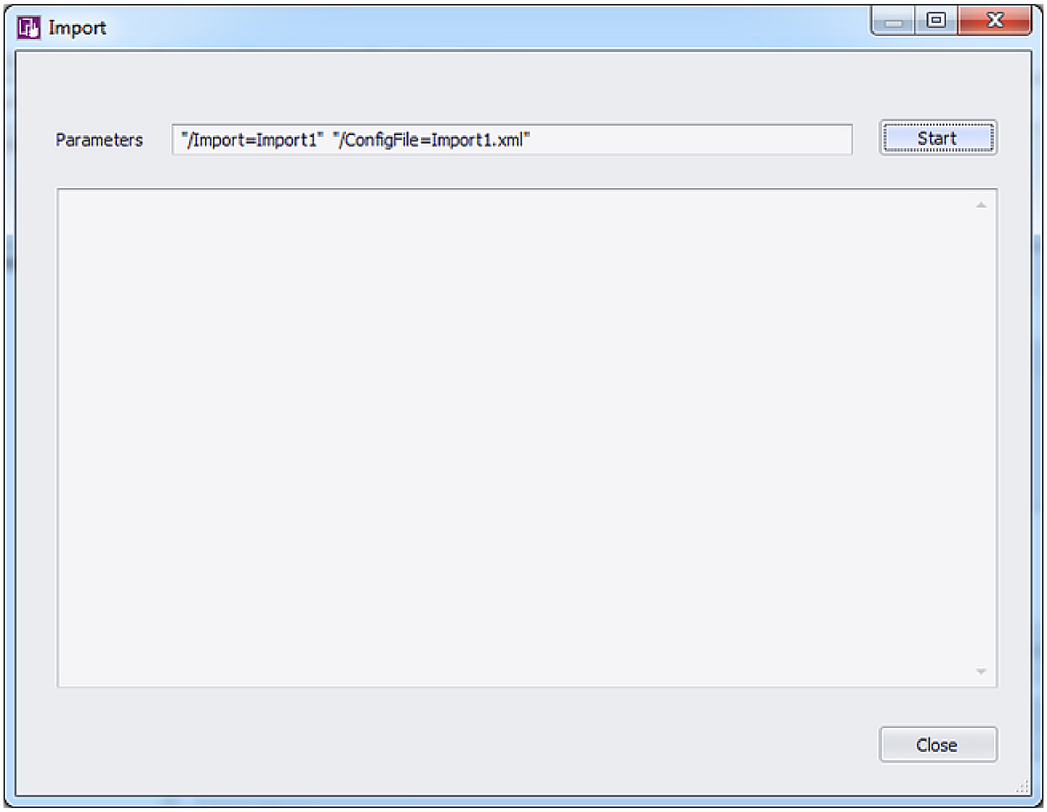
Always perform a full backup of the FlexNet Manager Suite database before performing an import. This will allow you to rollback the database to its original state if the imported data is corrupted or configuration of the import data is incorrect.

To trigger a data import through the Business Adapter Studio:

1. Make sure you have simulated the import and have confirmed that the import will perform the required tasks.
2. Back up the FlexNet Manager Suite database.
3. With the tab for the correct adapter selected, select the **Tools > Import** menu options.

   The **Import** page displays.

   Figure 19: Check the import parameters and then start the import

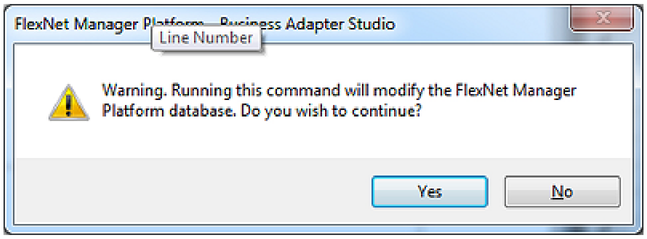


Parameters "/Import=Import1" "/ConfigFile=Import1.xml"

4. Click **Start**.

   A warning message displays.

   Figure 20: Confirm that you want to perform an import

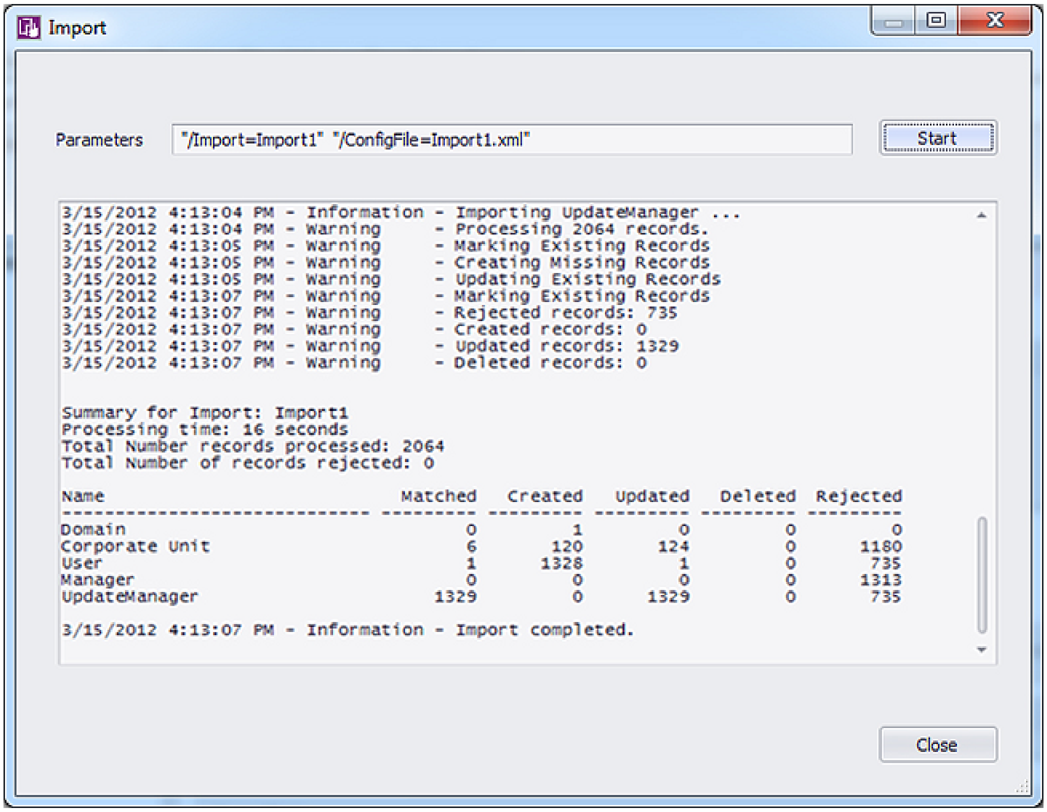


Warning. Running this command will modify the FlexNet Manager Platform database. Do you wish to continue?

5. Click **Yes** to confirm that you want to perform an import. (Alternatively, click **No** if you want to cancel the import.)

   The import is processed, and progress messages display on the window. When the import is complete, a summary of the records processed displays.

Figure 21: Review the results of the import process



"/Import=Import1" "/ConfigFile=Import1.xml"

```
3/15/2012 4:13:04 PM - Information - Importing UpdateManager ...
3/15/2012 4:13:04 PM - Warning     - Processing 2064 records.
3/15/2012 4:13:05 PM - Warning     - Marking Existing Records
3/15/2012 4:13:05 PM - Warning     - Creating Missing Records
3/15/2012 4:13:05 PM - Warning     - Updating Existing Records
3/15/2012 4:13:07 PM - Warning     - Marking Existing Records
3/15/2012 4:13:07 PM - Warning     - Rejected records: 735
3/15/2012 4:13:07 PM - Warning     - Created records: 0
3/15/2012 4:13:07 PM - Warning     - Updated records: 1329
3/15/2012 4:13:07 PM - Warning     - Deleted records: 0


Summary for Import: Import1
Processing time: 16 seconds
Total Number records processed: 2064
Total Number of records rejected: 0

Name                            Matched   Created   Updated   Deleted  Rejected
----------------------------- --------- --------- --------- --------- ---------
Domain                                0         1         0         0         0
Corporate Unit                        6       120       124         0      1180
User                                  1      1328         1         0       735
Manager                               0         0         0         0      1313
UpdateManager                      1329         0      1329         0       735

3/15/2012 4:13:07 PM - Information - Import completed.
```

6. Review the results. Check for any records rejected, and whether the correct number of records were processed as expected.
7. When you have finished reviewing the results, click **Close**.

> ***Tip:*** *If you expect to run this data import process more than once, it is best practise to set up a scheduled task to run the import on a regular basis. Use the Windows Scheduled Tasks system tool to create a scheduled task.*

To Review Past Imports in Business Adapter Studio

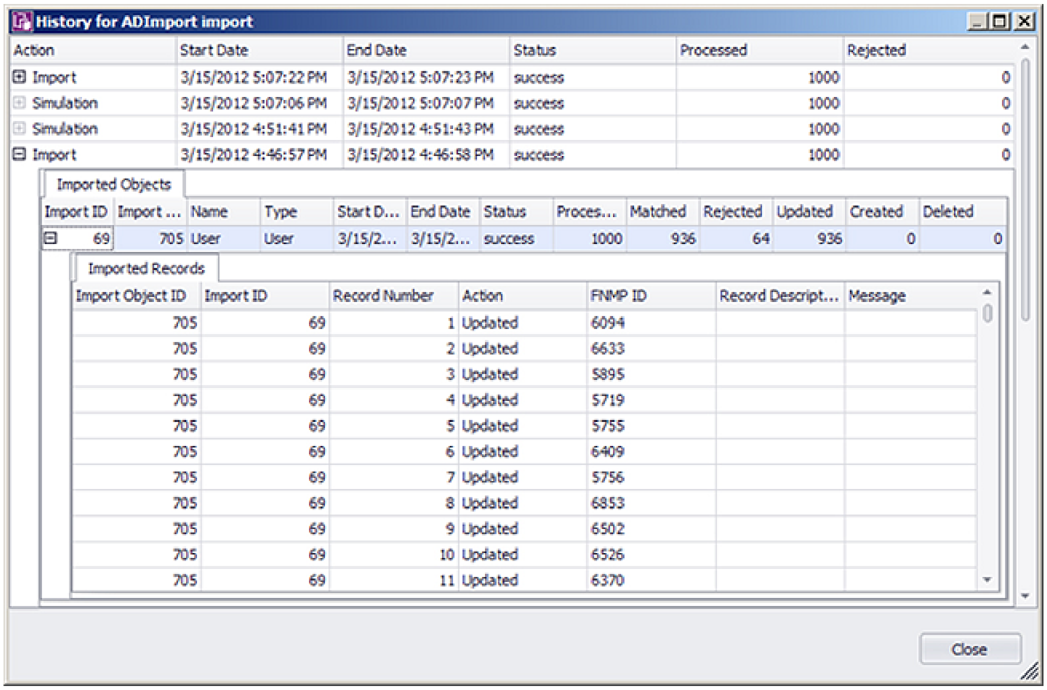
If you have run multiple imports of business information through Business Adapter Studio, you can review the history of your results. For regular imports, consider logging.

To review previous imports through the Business Adapter Studio:

1. With the appropriate tab displaying your adapter, select the **Tools > View History** menu options.

   The **History for** *`adapterName`* page displays.

Figure 22: A list of past imports and simulation imports displays



Action	Start Date	End Date	Status	Processed	Rejected
⊞ Import	3/15/2012 5:07:22 PM	3/15/2012 5:07:23 PM	success	1000	0
⊞ Simulation	3/15/2012 5:07:06 PM	3/15/2012 5:07:07 PM	success	1000	0
⊞ Simulation	3/15/2012 4:51:41 PM	3/15/2012 4:51:43 PM	success	1000	0
⊟ Import	3/15/2012 4:46:57 PM	3/15/2012 4:46:58 PM	success	1000	0

Import ID	Import ...	Name	Type	Start D...	End Date	Status	Proces...	Matched	Rejected	Updated	Created	Deleted
⊟ 69	705	User	User	3/15/2...	3/15/2...	success	1000	936	64	936	0	0

Import Object ID	Import ID	Record Number	Action	FNMP ID	Record Descript...	Message
705	69	1	Updated	6094		
705	69	2	Updated	6633		
705	69	3	Updated	5895		
705	69	4	Updated	5719		
705	69	5	Updated	5755		
705	69	6	Updated	6409		
705	69	7	Updated	5756		
705	69	8	Updated	6853		
705	69	9	Updated	6502		
705	69	10	Updated	6526		
705	69	11	Updated	6370		

2. To view details of the records imported (or simulated) during one of the actions listed on the page, expand the icon to the left of the import action.

   Details of each record imported or updated (or the simulation of that action) display.

3. When you have finished reviewing the history, click **Close** to close the window.

Setting Up Regular Imports (Connected Mode)

Many external data sources require regular imports to the compliance database to pick up later additions and modifications to the source data. For example, imports from your purchasing system must be repeated on a regular schedule to pick up new purchases that modify your overall license compliance.

To repeat the import using your new business adapter on a regular basis, create a standard Windows scheduled task on the application server.

Tip: *If you need to move your business adapter file from a development environment into your production environment, remember that you may need to modify the connection strings to suit.*

The default command line for your scheduled task has the following form:

```
InstallationPath\mgsbi.exe /Import=AdapterName /ConfigFile=XMLFileName
```

where the placeholders represent:

`InstallationPath`

The location where the Business Importer executable is installed on the compliance server. By default this is `C:\Program Files\ManagerSoft\DotNet\bin`.

`AdapterName`

The name displayed for your adapter in the structure tree of the Business Adapter Studio. For information about naming your adapter, see Renaming a Business Adapter (This name is also visible as the `Name` attribute for the `<Import>` element in the saved XML file.) Remember that if the adapter name includes spaces, you must enclose it in double quotation marks.

`XMLFileName`

The name under which you saved your XML file (see Saving Business Adapters). Once again, if the file name includes spaces, enclose it in double quotation marks.

Tip: *If you do not specify an XML file name, the default file `MGSBI.xml` is searched for the adapter.*