

Chapter 9

Integrating Data with the LDAP Directory

This chapter describes how developers can create data integrators that read data from a service provider's storage medium, and write the data in a format that complies with the SDX LDAP schema to a directory. The chapter contains the following sections:

- Overview of Data Integration on page 81
- Getting Help with Data Integration on page 83
- Installing the Data Integration Suite on page 84
- Planning Data Integration on page 84
- Developing Data Integrators on page 84
- Configuring Data Integrators on page 85
- Executing Data Integration on page 90
- Examples of Data Integrators on page 90

Overview of Data Integration

The SDX software uses data that complies with the SDX LDAP schema and that is stored in one of the supported directories. Service providers, however, may store data that they want to use for the SDX software in another storage medium, such as a database. You can develop data integrators that read your data from a storage medium, and write the data to a directory for use with the SDX software. A typical use of a data integrator is to read information about virtual private networks (VPNs) from your company's proprietary database and to write the data to a directory for use with the SDX software.

The data integration suite comprises a set of processors that perform different data management tasks. You can use combinations of these processors to achieve the data integration that you require. The processors use Extensible Markup Language (XML) documents to perform the data transfer. There are three types of processors:

- Readers, which read data from a storage medium and write the data to an XML document.
- Extensible Stylesheet Language Transformation (XSLT) transformers, which convert an XML document into a different XML document that another processor can use.
- Writers, which read data from an XML document in memory and write data to a storage medium.

Table 13 lists the processors, their functions, and associated document type definitions (DTDs). You can see the associated DTDs in the SDX software distribution in the folder *SDK/dtd/dataint* or on the Juniper Networks Web site at

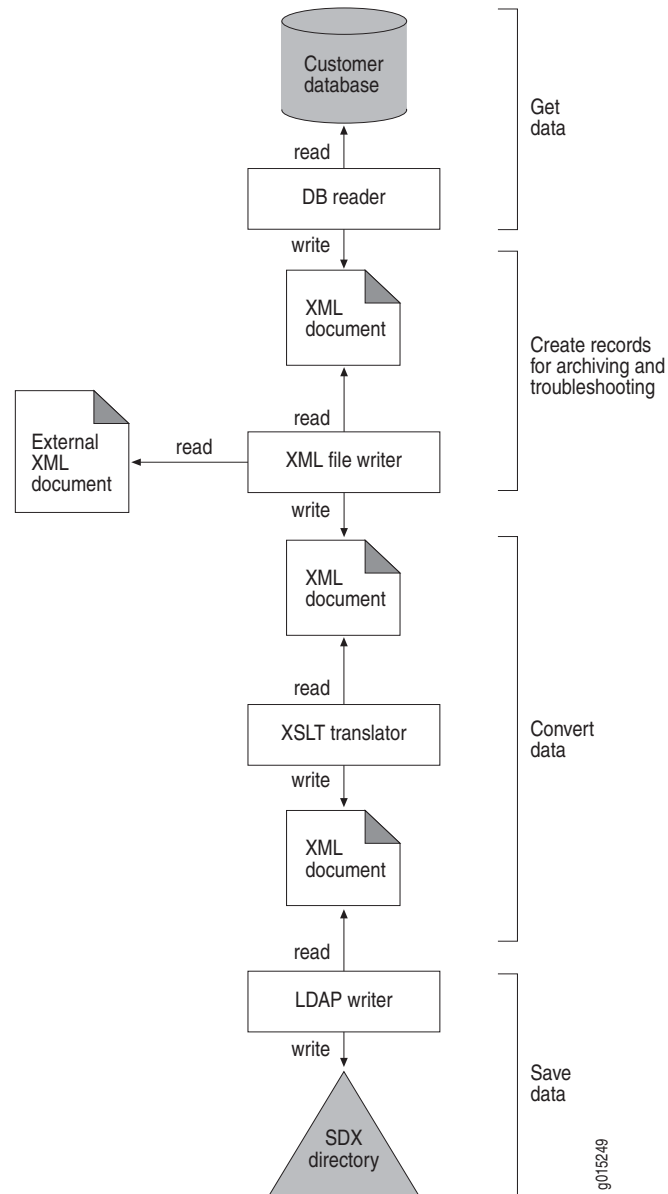
<http://www.juniper.net/techpubs/software/management/sdx>

Table 13: Processors for Data Managers

Processor	Function	Associated DTDs
Database Reader	Reads data from a database and writes the data to an XML document.	This processor does not have an associated DTD. Instead, it uses a DTD scheme with tags that correspond to the attribute names in the SQL queries that you write.
LDAP Reader	Reads data from an LDAP-compliant directory and writes the data to an XML document.	This processor uses two DTDs: <ul style="list-style-type: none"> ■ The input DTD, <i>LDAPReader_input.dtd</i>, describes a query to a directory. ■ The output DTD, <i>LDAPReader_output.dtd</i> describes the result of a query to the directory.
XML File Reader	Reads data from an XML document and passes the data to the next processor.	None
Enterprise Audit File Reader	Reads data from the Enterprise Service Portal audit plug-in log and writes the data to an XML document with a DTD specific to the enterprise audit plug-in log.	This processor uses the DTD <i>EntAuditFileReader_output.dtd</i> , which describes the result of a query to an enterprise audit log.
XML File Writer	Reads data from an XML document, writes the data to an external XML document, and returns the original XML document. Use this processor to create external XML documents of data either for record-keeping purposes or for troubleshooting the data integration.	None
XSLT Translator	Takes the XML document it receives from the preceding processor and writes it to an XML document that another processor can read. For this processor, you must customize an XSLT file that maps the input XML document to the output XML document.	None
LDAP Writer	Reads data from an XML document and writes the data to an LDAP directory.	This processor uses the DTD <i>LDAPWriter_input.dtd</i> , which describes a set of directory updates.

Figure 9 illustrates how you can use a combination of the processors to transfer data from your database to a directory that contains SDX data. For more detailed examples, see *Examples of Data Integrators* on page 90.

Figure 9: Transferring Data from a Database to the SDX Directory



Getting Help with Data Integration

Planning data integration and developing data integrators are tasks that should be undertaken by developers with knowledge of XML, XSLT, databases, and directories. For assistance with these tasks, consult Juniper Networks Professional Services.

Installing the Data Integration Suite

For information about installing the data integration suite, see *SDX Getting Started Guide, Chapter 5, Installing the SDX-300 Software*. After you have installed the data integration package, copy the Java Database Connectivity (JDBC) drivers for any databases from which you will extract data to the folder *opt/UMC/datint/lib/*.

Planning Data Integration

Before you develop data integrators, create a plan for the data integration. To do so:

1. Determine the data that you want to transfer, the sources of the data, and the type of transfer that you want to perform.

For example, you might want to transfer data about subscribers, networks, and VPNs from your database to the SDX directory. You might also want to create XML documents of the original data that you can use for troubleshooting as you create the data manager.

2. Determine how to divide the transfer process into smaller data transfers that limit the size of the query result and minimize the time that the LDAP Writer spends interacting with the directory.

For example, you might create three data integrators, each of which manages data for one of the following areas:

- Subscribers
 - Network
 - VPNs
3. Identify the processors that you need to perform the data transfer, and the properties that you will use for each processor.

For a detailed description of each processor and its properties, see the online documentation in the SDX software distribution in *SDK/doc/dataint*.

Developing Data Integrators

To develop a data integrator:

1. For each XSLT transformer that you use, create an XSLT file that maps the structure of the input XML document to an output XML document that the next processor in the chain can read.

You can view examples of XSLT files in the folder */opt/UMC/datint/xslt*. For more information about these examples, see *Examples of Data Integrators* on page 90.

2. If you are using the Database Reader processor, write an SQL query that the processor uses to obtain information from the database.

3. Create a property file that defines the properties for the data integrator (see *Configuring Data Integrators* on page 85).
4. Create a script that calls the data integrator.
5. (Optional) Configure a utility, such as a crontab file, to run this script at a defined time.

The script or crontab file can run the data integrator either once with a single property file or multiple times with different property files. For an example of a script that calls a data integrator, see the file *vpndatamgt* in */opt/UMC/datint/etc*.

Configuring Data Integrators

To configure a data integrator, you must create a property file that contains specific properties for the overall data transfer and for the individual data processors. When you create a property file, you must:

1. Define logging properties by using the standard property names and values for SDX logging. To define the logging properties, use the following format:

Logger.<groupName>.<propertyName>=<value>

- <groupName> —Name of group for this log
- <propertyName> —Name of property
- <value> —Value of property

For detailed information about configuring SDX logging properties, see *SDX Monitoring and Troubleshooting Guide, Chapter 2, Configuring Logging for SDX Components*.

2. Define properties for the individual processors. The properties that you must or can configure depend on the particular processor. To define properties for the processors, use the following format:

Processor.<processorName>.<propertyName>=<value>

- <processorName> —Name of processor that you define; each processor in a property file must have a unique name
- <propertyName> —Name of property; may comprise several text strings separated by dots
- <value> —Value of property

To define properties for each processor, see:

- *Defining Properties for the Database Reader* on page 86
- *Defining Properties for the LDAP Reader* on page 87
- *Defining Properties for the XML File Reader* on page 88
- *Defining Properties for the Enterprise Audit File Reader* on page 88
- *Defining Properties for the XML File Writer* on page 89

- *Defining Properties for the XSLT Translator* on page 89
- *Defining Properties for the LDAP Writer* on page 89

For detailed information about each processor, see the online documentation in the SDX software distribution in *SDK/doc/dataint*.

3. Define the order in which the processors are called. To define the order in which the processors will be executed, enter one statement in the following format:

`Processor.chain=<comma-separated list of names of processors in order>`

For example:

`Processor.chain=dbreader,toldap,xmlfilewriter,ldapwriter`

Defining Properties for the Database Reader

You must define the following properties for this processor:

- Class of the processor
- Data that you want to read from the database

`Processor.dbreader.dbQuery=SELECT <sqlQuery>`

 - `<sqlQuery >` —SQL query that summarizes and processes data from the database
- Class of the JDBC driver

`Processor.dbreader.driverClass=<driverClass>`

 - `< driverClass >` —Class of JDBC driver
- URL of the database

`Processor.dbreader.dbURL=<databaseURL>`

 - `< databaseURL >` —URL of the database
- Username and password for logging into the database

`Processor.dbreader.user=<username>`
`Processor.dbreader.password=<password>`

 - `< username >` —Username that the database uses to authenticate the processor
 - `< password >` —Password that the database uses to authenticate the processor
- Output format, Document Object Model (DOM) for the query result

`Processor.dbreader.out=dom`

- Inclusion of data in the XML document that the processor returns

```
Processor.dbreader.genData=true
```

- Any optional properties that you require

For information about optional properties for this processor, see the online documentation in the SDX software distribution in *SDK/doc/dataint*.

The following example is a property file for this processor:

```
# Database Reader
Processor.dbreader.class=net.juniper.smgmt.ent.datamgt.reader.DBReader
Processor.dbreader.driverClass=org.gjt.mm.mysql.Driver
Processor.dbreader.dbURL=jdbc:mysql://127.0.0.1:3306/vpn
Processor.dbreader.user=admin
Processor.dbreader.password=secret
Processor.dbreader.genData=true
Processor.dbreader.out=dom

# The SQL query
Processor.dbreader.dbQuery=SELECT vpn_ownership.vpn_id,
vpn_ownership.vpn_owner,vpn_sites.router_name,vpn_sites.interface_nameFROM
vpn_ownership, vpn_sites where vpn_ownership.vpn_id=vpn_sites.vpn_id

# XML element names
Processor.dbreader.ename.database=database
#Processor.dbreader.ename.record=record
```

Defining Properties for the LDAP Reader

This processor obtains the query it performs as an XML document from the previous processor in the chain, and you do not need to define the query. You must, however, define the following properties for this processor:

- Name of the class of the processor
- DES properties in the format

```
Processor.<processorName>.<desProperty>=<value>
```

- `<processorName >` —Name of processor that you define; each processor in the same property file must have a unique name
- `<desProperty >` —Name of the DES property
- `<value >` —Value of the DES property

For information about DES properties and values, see *SDX Getting Started Guide, Chapter 14, Distributing Directory Changes to SDX Components*.

- Any optional properties that you require

For information about optional properties for this processor, see the online documentation in the SDX software distribution in *SDK/doc/dataint*.

The following example is a property file for this processor:

```
Processor.Ldapreader.class=net.juniper.smgmt.ent.datamgt.reader.LDAPReader
Processor.Ldapreader.java.naming.provider.url = ldap://127.0.0.1/
Processor.Ldapreader.java.naming.security.principal = cn=umcadmin,o=umc
Processor.Ldapreader.java.naming.security.credentials = admin123
Processor.Ldapreader.continuous=true
Processor.Ldapreader.java.naming.provider.url = ldap://127.0.0.1/
```

Defining Properties for the XML File Reader

You must define the following properties for this processor:

- Name of the class of the processor
- In the following format, the XML document from which this processor reads data:

```
Processor.<processorName>.XMLFileName= <path>
```

- `< processorName >` —Name of the processor that you define; each processor in the same property file must have a unique name
- `< path >` —Path to XML document relative to the folder `/opt/UMC/datint`

The following example is a property file for this processor:

```
Processor.xmlfilereader.class=net.juniper.smgmt.ent.datamgt.reader.XMLFileReader
Processor.xmlfilereader.XMLFileName=var/log/dbout.xml
```

Defining Properties for the Enterprise Audit File Reader

You must define the following properties for this processor:

- Name of the class of the processor
- In the following format, the log file from which this processor reads data:

```
Processor.<processorName>.auditFileName=<path>
```

- `< processorName >` —Name of the processor that you define; each processor in the same property file must have a unique name
- `< path >` —Path to the XML document relative to the folder `/opt/UMC/datint`
- Any optional properties that you require

For information about optional properties for this processor, see the online documentation in the SDX software distribution in `SDK/doc/dataint`.

The following example is a property file for this processor:

```
Processor.auditfilereader.class=net.juniper.smgmt.ent.datamgt.reader.EntAuditFileReader
Processor.auditfilereader.auditFileName=ent_audit.log
Processor.auditfilereader.filter=(Action=Unexport-VPN)
```

Defining Properties for the XML File Writer

You must define the following properties for this processor:

- Name of the class of the processor
- In the following format, the XML document to which the processor writes data:

```
Processor.xmlfilewriter.XMLFileName=<path>
```

- < path > —Path to the XML document relative to the folder */opt/UMC/datint*

The following example is a property file for this processor:

```
Processor.xmlfilewriter.class=net.juniper.smgmt.ent.datamgt.filter.XMLFileWriter
Processor.xmlfilewriter.XMLFileName=var/log/ldapout.xml
```

Defining Properties for the XSLT Translator

You must define the following properties for this processor:

- Name of the class of the processor
- In the following format, the XSLT file that the processor uses

```
Processor.<processorName>.XSLTFileName=<path>
```

- < processorName > —Name of the XSLT translator
- < path > —Path to XSLT file relative to the folder */opt/UMC/datint*

The following example is a property file for this processor:

```
Processor.toabstract.class=net.juniper.smgmt.ent.datamgt.filter.XSLTTranslator
Processor.toabstract.XSLTFileName=xslt/vpn.xslt
```

Defining Properties for the LDAP Writer

You must define the following properties for this processor:

- Name of the class of the processor
- DES properties in the format

```
Processor.ldapwriter.<desProperty>=<value>
```

- < desProperty > —Name of the DES property
- < value > —Value of the DES property

For information about DES properties and values, see *SDX Getting Started Guide, Chapter 14, Distributing Directory Changes to SDX Components*.

- Any optional properties that you require

For information about optional properties for this processor, see the online documentation in the SDX software distribution in *SDK/doc/dataint*.

The following example is a property file for this processor:

```
Processor.Ldapwriter.class=net.juniper.smgmt.ent.datamgt.filter.LDAPWriter
Processor.Ldapwriter.java.naming.provider.url = ldap://127.0.0.1/
Processor.Ldapwriter.java.naming.security.principal = cn=umcadmin,o=umc
Processor.Ldapwriter.java.naming.security.credentials = admin123
Processor.Ldapwriter.updateRateLimit=3
Processor.Ldapwriter.continuous=true
```

Executing Data Integration

To execute data integration, run the script that calls the property files, or configure a utility, such as a crontab file, to run this script at a defined time.

Examples of Data Integrators

After you install the data integration suite, you can access two data integrators that we have developed:

- VPN Directory Updater
- VPN Subscription Deactivator

The following sections describe each of these data integrators. You can also examine the following files to see how these data integrators were developed.

- Property files in */opt/UMC/datint/etc*
- XSLT files in */opt/UMC/datint/xslt*
- The script **vpndatamgt** in */opt/UMC/datint/etc*, which calls both these data integrators

Example: VPN Directory Updater

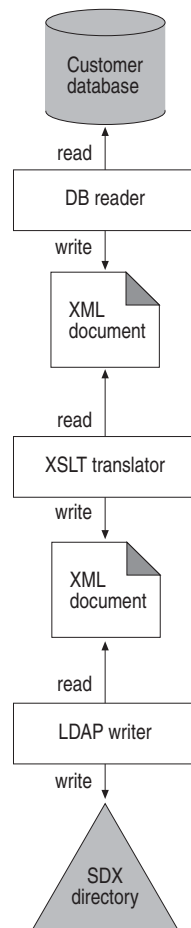
VPN Directory Updater is a sample data integrator that reads data about VPNs from a database and writes to a directory the data in a format that meets the SDX LDAP schema. If you want to use VPN Directory Updater, you must customize it for your specific application. At the very least, you need to customize the SQL queries for your database.

VPN Directory Updater works as follows:

1. Database Reader submits SQL queries to a database, obtains the result of the query, and converts the result to an XML document.
2. XSLT Translator takes the XML document produced by Database Reader and converts it to an XML document that describes a set of directory updates.
3. LDAP Writer uses the XML document generated by XSLT Translator to update the directory.

Figure 10 illustrates this process.

Figure 10: VPN Directory Updater Operation



g015275

Example: VPN Subscription Deactivator

If an IT manager cancels the export of a VPN at the same time that an extranet client activates a subscription to this VPN, there is a remote possibility that the Enterprise Manager portal will maintain the active, but invalid, subscription. VPN Subscription Deactivator deactivates this type of invalid VPN subscription.

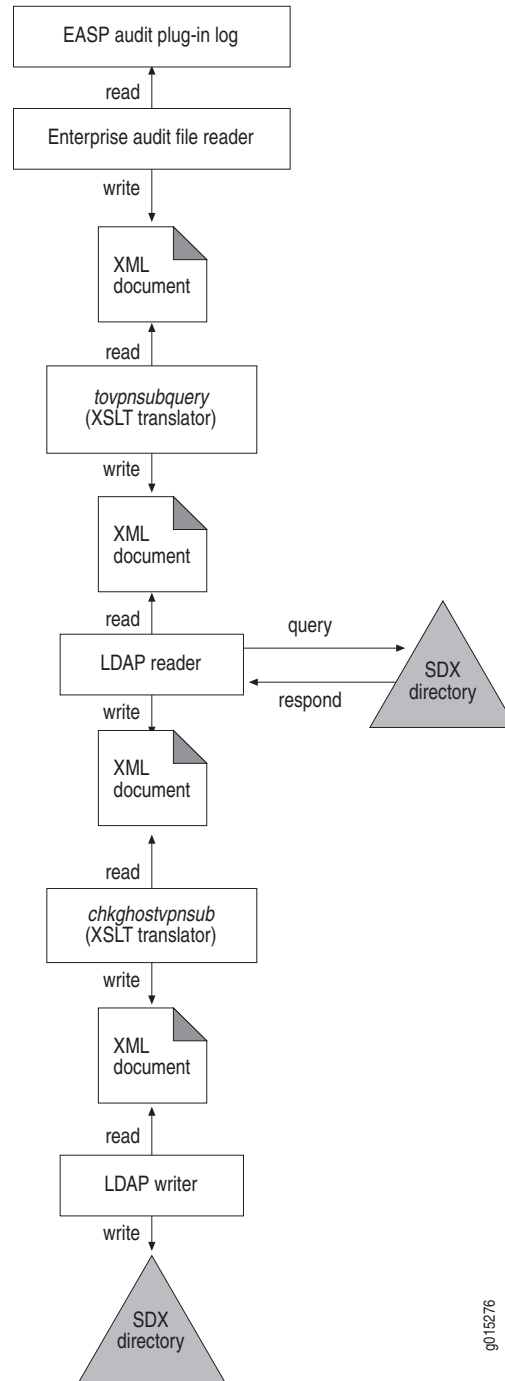
You can use VPN Subscription Deactivator without modifications. For information about using this data integrator, see *SDX Subscribers and Subscriptions Guide, Chapter 14, Adding VPNs from JUNOS Routing Platforms*.

VPN Subscription Deactivator works as follows:

1. Enterprise Audit File Reader finds events of type `unexport-vpn` in the log for the Enterprise Audit Plug-In and converts the events to an XML document.
2. The XSLT Translator `tovpnsubquery` performs the following actions:
 - a. Reads the XML document from the Enterprise Audit File Reader and uses it to identify extranet clients for whom export of a VPN was canceled.
 - b. Uses the XSLT file `tovpnsubquery.xslt` to generate LDAP queries to obtain subscriptions, imported extranets, and VPNs owned for these extranet clients.
 - c. Writes the queries to an XML document.
3. LDAP Reader performs the following actions:
 - a. Reads the XML document from the XSLT Translator `tovpnsubquery`.
 - b. Submits the queries to the directory.
 - c. Obtains the results of the queries from the directory.
 - d. Writes the results to an XML document.
4. The XSLT Translator `chkghostvpnsub` performs the following actions:
 - a. Reads the XML document from the LDAP Reader.
 - b. Uses the XSLT file `chkghostvpnsub.xslt` to find in the XML document VPN subscriptions that are still active even though the export of the VPN has been canceled.
 - c. Generates an XML document that contains LDAP updates to deactivate the invalid subscriptions.
5. LDAP Writer uses the XML documents generated by Enterprise Audit File Reader and LDAP Reader to update the directory.

Figure 11 illustrates this process.

Figure 11: VPN Subscription Activator



0015276

