opentext™

OpenText™ Identity Manager CE 24.4 (v4.10)

Security Guide

January 2025

Legal Notice

Copyright 2009 - 2024 Open Text.

The only warranties for products and services of Open Text and its affiliates and licensors (“Open Text”) are as may be set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Open Text shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Contents

About this Book and the Library **5**

1 Overview for Securing OpenText Identity Manager Components **7**

2 Security Best Practices **9**

Using SSL . . . 9
Securing Directory Access . . . 9
Granting Task-Based Access to Drivers and Driver Sets . . . 10
Understanding OpenText Identity Manager Communication . . . 12
Managing Passwords . . . 13
Creating Strong Password Policies . . . 14
Securing ActiveMQ Communication . . . 15
Configuring a Whitelist of Target URLs . . . 15
Configuring a Whitelist of Target URLs in Identity Applications . . . 16
Configuring a Whitelist of Target URLs in OSP . . . 16
Preventing Clickjacking Attacks . . . 17
Rejecting Client-initiated SSL Renegotiation on Windows . . . 18
Using Tomcat connectionTimeout Attribute to Defend Against Denial-of-Service Attack . . . 19
Securing Connected Systems . . . 19
Password Generation . . . 20
Designer for OpenText Identity Manager . . . 20
Industry Best Practices for Security . . . 21
Tracking Changes to Sensitive Information . . . 21
Using Designer to Log Events . . . 21
Establishing a Security Equivalent User . . . 24
Securing Communication Between ActiveMQ and Identity Applications . . . 25
Creating a Keystore and a Truststore . . . 26
Enabling SSL for ActiveMQ . . . 27
Enabling SSL for the Identity Applications Client . . . 27
Securing Cloud Deployment of OpenText Identity Manager Components . . . 27
Securing the Information with is-sensitive Attribute . . . 28
Configuring Stronger Ciphers for SSL Communication . . . 29
Prerequisites . . . 30
Configuring the Settings for Suite B Mode . . . 30
Enabling Stronger Ciphers for SSL Communication . . . 32
Verifying the Suite B Settings . . . 32

About this Book and the Library

The *Security Guide* provides information about security best practices you might want to implement in your environment.

Intended Audience

This book provides information for individuals responsible for understanding administration concepts and implementing a secure, distributed administration model.

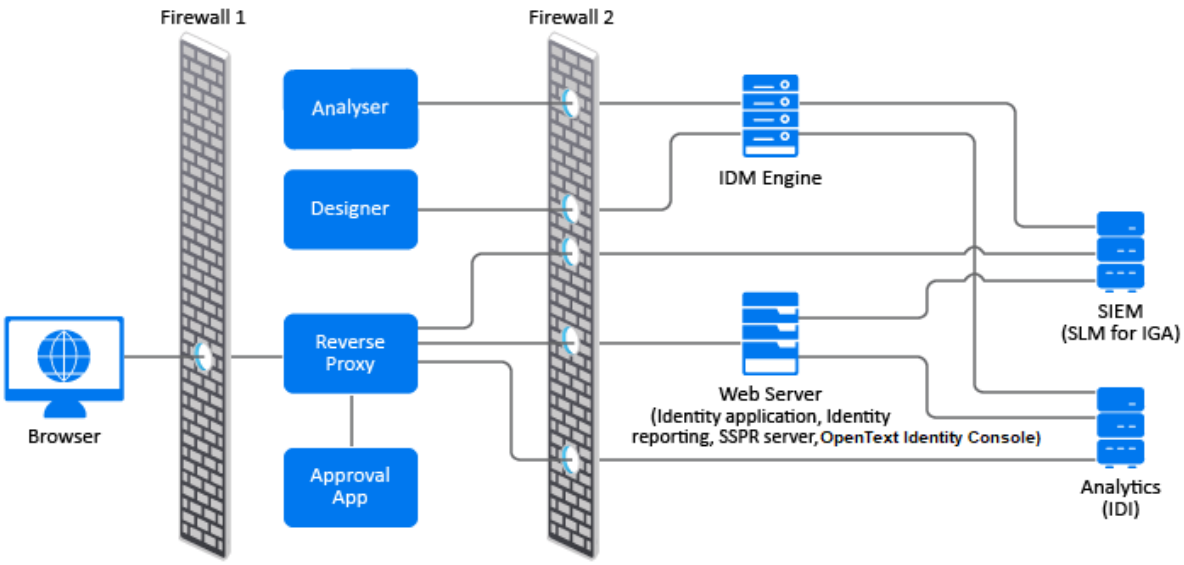
Other Information in the Library

For more information about the library for OpenText Identity Manager, see the OpenText Identity Manager documentation website.

1 Overview for Securing OpenText Identity Manager Components

Most organizations have their identity data stored on multiple systems. In this case, managing identities and monitoring user activity on physical and virtual environments are important. OpenText Identity Manager provides an automated environment to synchronize the identity data across multiple connected systems. It also ensures that users have access to the resources required for their jobs. As it creates and stores a lot of user data, ensuring that the data is protected from unauthorized use, access, and modification is important. To protect your data, OpenText recommends a few best practices for deploying and securing the OpenText Identity Manager system.

Figure 1-1 *Securing the Product Components between Firewalls*



First Firewall: If you place a firewall between browsers and Analyser, Designer, Reverse Proxy, and Approval App, you need to open ports so that browsers can communicate with these components. For more information about ports, see "Understanding OpenText Identity Manager Communication" on page 12.

Second Firewall: The second firewall separates IDM Engine, web servers, SIEM and Analytics Server from Analyser, Designer, Reverse Proxy and Approval App. For more information about security best practices, see Chapter 2, "Security Best Practices," on page 9.

2 Security Best Practices

The following sections provide information about securing the OpenText Identity Manager system:

- "Using SSL" on page 9
- "Securing Directory Access" on page 9
- "Understanding OpenText Identity Manager Communication" on page 12
- "Managing Passwords" on page 13
- "Creating Strong Password Policies" on page 14
- "Securing ActiveMQ Communication" on page 15
- "Configuring a Whitelist of Target URLs" on page 15
- "Preventing Clickjacking Attacks" on page 17
- "Rejecting Client-initiated SSL Renegotiation on Windows" on page 18
- "Using Tomcat connectionTimeout Attribute to Defend Against Denial-of-Service Attack" on page 19
- "Securing Connected Systems" on page 19
- "Designer for OpenText Identity Manager" on page 20
- "Industry Best Practices for Security" on page 21
- "Tracking Changes to Sensitive Information" on page 21
- "Establishing a Security Equivalent User" on page 24
- "Securing Communication Between ActiveMQ and Identity Applications" on page 25
- "Securing Cloud Deployment of OpenText Identity Manager Components" on page 27
- "Securing the Information with is-sensitive Attribute" on page 28
- "Configuring Stronger Ciphers for SSL Communication" on page 29

Using SSL

Enable SSL for all transports, where it is available. Enable SSL for communication between the OpenText Identity Manager engine and Remote Loader and between the OpenText Identity Manager engine or Remote Loader and the connected systems. For information, see "Creating a Secure Connection to the OpenText Identity Manager Engine" in the *OpenText™ Identity Manager CE 24.4 (v4.10) Driver Administration Guide*.

If you don't enable SSL, you are sending sensitive information such as passwords in clear text.

Securing Directory Access

Make sure that you secure access to Identity Vaults and to products' objects.

Physical Security: Protect access to the physical location of the servers where an Identity Vault is installed.

File System Access: The security of the file system is critical to ensuring the security of the system as a whole. Verify that the directories containing OpenText eDirectory, the OpenText Identity Manager engine, and the Remote Loader are accessible only to the appropriate administrators.

There is an issue with the file system when the Remote Loader is installed on a Windows 2000 server. For more information, see TID 3243550, Securing a Remote Loader Install on a Microsoft Windows 2000 Server (https://www.novell.com/support/kb/doc.php?id=3243550).

The OpenText Identity Manager files and directories have permissions that specify who and what can read, write, modify, and access them. This is important because OpenText Identity Manager may need access to write to files in your file system to enable certain functions.

The contains the minimum required permissions for different directories:

Table 2-1 *Minimum File and Directory Permissions*

Path	Directory	Minimum Permission Required
/var/opt/novell/	eDirectory	755
/var/opt/novell/	dirxml	750
/var/opt/novell/	idm	750

To see the permissions set for a file, open the command line and type: `ls -l`.

For security requirements, OpenText recommends that you make a note of these permissions.

Access Rights: OpenText Identity Manager requires administrative rights to create objects and configure drivers. Monitor and control who has rights to create or modify the following:

- An OpenText Identity Manager driver set
- An OpenText Identity Manager driver
- Driver configuration objects (filters, style sheets, policies), especially policies that are used for password retrieval or synchronization
- Password policy objects (and the OpenText Identity Console task for editing them), because they control which passwords are synchronized to each other, and which Password Self-Service options are used

Granting Task-Based Access to Drivers and Driver Sets

In addition to the OpenText eDirectory standard object-based access controls, OpenText Identity Manager lets you assign trustee rights to perform only certain tasks on an OpenText Identity Manager driver, rather than just granting full Supervisor rights to the driver object. For example, you can assign trustee rights so that one user can only configure the driver object (create and modify object properties), while another user can only start and stop the driver.

OpenText Identity Manager provides the following driver object attributes that enable role-based access:

Attribute	Description
DirXML-AccessRun	Start and stop OpenText Identity Manager drivers and jobs
DirXML-AccessMigrate	Manage migration operations into the Identity Vault
DirXML-AccessSubmitCommand	Manage the driver's pass-through commands
DirXML-AccessCheckObjectPassword	Manage the driver's check object password commands
DirXML-AccessConfigure	Manage the driver's configuration and job configuration
DirXML-AccessManage	View and modify the driver's cache file contents

Setting trustee rights to these attributes grants access to the associated OpenText Identity Manager verbs and sub-verbs. Read access lets users view state (get verb state), and Write access lets users modify or change state (set verb state.) For example, granting Read access to a driver object's DirXML-AccessRun attribute lets the user get the driver state (started or stopped.) Granting Write access lets the user set the driver state (change from started to stopped, or vice versa.)

The goal of providing this attribute-based access to driver tasks is to let you create well-defined administrative roles, perhaps using the OpenText eDirectory Administrative Role object, that let users perform certain management tasks without exposing all management functionality. Creating these roles can go beyond providing access to the DirXML-Access attributes described above and can include access rights to other attributes, as well as access to other OpenText Identity Manager objects. The following examples demonstrate the flexibility available for creating administrative roles:

Start/Stop Driver Admin: This administrative role lets the assigned user start and stop all drivers in a given driver set. It requires the following access rights:

- Browse rights to the Driver Set object
- Read and Write access, with inheritance, to the DirXML-AccessRun attribute of the Driver Set object

Driver Admin: This administrative role lets the assigned user manage a single Driver object. It requires the following access rights:

- Browse and Create rights to the Driver object
- Read and Write access to [All Attribute Rights] in the Driver object

NOTE: Make sure the rights are inherited so the driver Admin can also manage the driver's policy objects.

For more information about using OpenText Identity Console to grant OpenText eDirectory access rights is available in the OpenText Identity Console Administration Guide (https://www.netiq.com/documentation/identity-console/identity_console-admin/data/bookinfo.html).

Understanding OpenText Identity Manager Communication

The product components use different ports for proper communication among them.

NOTE: If a default port is already in use, ensure that you specify a different port for the component.

Port Number	Component Computer	Port Use
389	Identity Vault	Used for LDAP communication in clear text with OpenText Identity Manager components
465	Identity Reporting	Used for communication with the SMTP mail server
524	Identity Vault	Used for NetWare Core Protocol (NCP) communication
636	Identity Vault	Used for LDAP with TLS/SSL communication with OpenText Identity Manager components
5432	Identity Applications	Used for communication with the identity applications database
7707	Identity Reporting	Used by the Managed System Gateway driver to communicate with the Identity Vault
8000	Remote Loader	Used by the driver instance for TCP/IP communication **NOTE:** Each instance of the Remote Loader should be assigned a unique port.
8005	Identity Applications	Used by Tomcat to listen for shutdown commands
8009	Identity Applications	Used by Tomcat for communication with a web connector using the AJP protocol instead of HTTP
8028	Identity Vault	Used for HTTP clear text communication with NCP communication
8030	Identity Vault	Used for HTTPS communication with NCP communication
8080	Identity Applications	Used by Tomcat for HTTP clear text communication
8090	Remote Loader	Used by the Remote Loader to listen for TCP/IP connections from the remote interface shim **NOTE:** Each instance of the Remote Loader should be assigned a unique port.
8109	Identity Applications	Applies only when using the integrated installation process Used by Tomcat for communication with a web connector using the AJP protocol instead of HTTP
8180	Identity Applications	Used for HTTP communications by the Tomcat application server on which the identity applications run

Port Number	Component Computer	Port Use
8443	Identity Applications	Used by Tomcat for HTTPS (SSL) communication or redirecting requests for SSL communication
8543	Identity Applications	*Not listening, by default* Used by Tomcat to redirect requests that require SSL transport when you do not use TLS/SSL protocol
9009	OpenText Identity Console	Used by Tomcat for MOD_JK
5432	Identity Reporting	Used for the PostgreSQL database Sentinel
45654	User Application	Used by the server on which the database for the identity applications are installed to listen for communications, when running Tomcat with a cluster group

Managing Passwords

When you choose to exchange information between connected systems, you should take precautions to make sure that the exchange is secure. This is especially true for passwords.

- The Password Hint attribute (nsimHint) is publicly readable, to allow unauthenticated users who have forgotten a password to access their own hints. Password Hints can help reduce help desk calls.

  For security, Password Hints are checked to make sure that they do not contain the user's actual password. However, a user could still create a Password Hint that gives too much information about the password.

  To increase security when using Password Hints:

  - Allow access to the nsimHint attribute only on the LDAP server used for Password Self-Service.
  - Require that users answer Challenge Questions before receiving the Password Hint.
  - Remind users to create Password Hints that only they would understand. The Password Change Message in the password policy is one way to do this. See “Adding a Password Change Message” in the *Password Management Administration Guide*.

  If you choose not to use Password Hint at all, make sure you don't use it in any of the password policies. To prevent Password Hints from being set, you can go a step further and remove the Hint Setup gadget completely, as described in “Disabling Password Hint by Removing the Hint Gadget” in the *Password Management Administration Guide*.

- Challenge Questions are publicly readable, to allow unauthenticated users who have forgotten a password to authenticate another way. Requiring Challenge Questions increases the security of Forgotten Password Self-Service, because a user must prove his or her identity by giving the correct responses before receiving a forgotten password or a Password Hint, or resetting a password.

  The intruder lockout setting is enforced for Challenge Questions, so the number of incorrect attempts an intruder could make is limited.

However, a user could create Challenge Questions that hold clues to the password. Remind users to create Challenge Questions and Responses that only they would understand. The Password Change Message in the password policy is one way to do this. See “Adding a Password Change Message” in the *Password Management Administration Guide*.

- For security, the Forgotten Password actions of **E-mail password to user** and **Allow user to reset password** are available only if you require the user to answer Challenge Questions.
- A security enhancement was added to NMAS 2.3.4 regarding Universal Passwords changed by an administrator. It works basically the same way as the feature previously provided for NDS Password.

  If an administrator changes a user's password, such as when creating a new user or in response to a help desk call, the password is automatically expired if you have enabled the setting to expire passwords in the password policy. The setting in the password policy is in Advanced Password Rules, named **Number of days before password expires (0-365)**. For this particular feature, the number of days is not important, but the setting must be enabled.
- You are recommended to use password-ref GCV for passwords.

Creating Strong Password Policies

Password policy objects are publicly readable to allow applications to check whether passwords are compliant. This means that an unauthenticated user could query an Identity Vault and find out what password policies are in place. If the password policies require users to create strong passwords, this should not pose a risk, as noted in “Create Strong Password Policies” in the *Password Management Administration Guide*.

Password Synchronization lets you simplify user passwords and reduce help desk costs. Bidirectional password synchronization lets you share passwords among OpenText eDirectory and connected systems in multiple ways, as described in the scenarios in the *OpenText™ Identity Manager CE 24.4 (v4.10) Password Management Guide*.

Using Universal Password and password policies allows you to enforce strong password syntax requirements for users. Use the Advanced Password Rules in password policies to define your organization’s best practices for passwords. The Advanced Password Rules features let you manage password syntax by using either syntax or the Microsoft Complexity Policy. For more information, see “Managing Passwords by Using Password Policies” in the *Password Management Administration Guide*.

For example, using NetIQ password syntax options, you can require user passwords to comply with rules such as the following:

- Requiring unique passwords.

  You can prevent users from reusing passwords, and control the number of passwords the system should store in the history list for comparison
- Requiring a minimum number of characters in the password.

  Requiring longer passwords is one of the best ways to make passwords stronger.
- Requiring a minimum number of numerals in the password.

  Requiring at least one numeric character in a password helps protect against “dictionary attacks,” in which intruders try to log in using words in the dictionary.

- Excluding passwords of your choice.

  You can exclude words that you consider to be security risks, such as the company name or location, or the words “test” or “admin.” Although the exclusion list is not meant to import an entire dictionary, the list of words you exclude can be quite long. Just keep in mind that a long list of exclusions makes login slower for your users. A better protection from dictionary attacks is to require numerals or special characters.

Keep in mind that you can create multiple password policies if you have different password requirements in different parts of the tree. You can assign a password policy to the whole tree, a partition root container, container, or even an individual user. (To simplify administration, we recommend that you assign password policies as high up in the tree as possible.)

In addition, you can use intruder lockout. This feature lets you specify how many failed login attempts are allowed before an account is locked. This is a setting on the parent container instead of in the password policy. See “Managing User Accounts” in the *OpenText eDirectory Administration Guide*.

Securing ActiveMQ Communication

OpenText Identity Manager uses ActiveMQ for supporting messaging service in the following components:

- JDBC Fan-Out driver, between the Fan-Out driver shim and ActiveMQ and between ActiveMQ and the Fan-Out agent

  For more information, see Securing Fanout Driver Communication in the *OpenText Identity Manager Driver for JDBC Fanout Implementation Guide*.

- User Application

OpenText recommends using Secure Socket Layer (SSL) protocol for ActiveMQ communication. By default, SSL protocol is not configured in the components interacting with ActiveMQ. You must manually configure it.

Configuring a Whitelist of Target URLs

URL redirection, which many applications and services require, inherently brings in security risks. While redirecting, the request can be tampered to redirect users to an external, malicious site. To prevent such issues, you can configure a list of permissible URLs in OSP configuration. This restricts redirection only to the configured URLs. For example, when an authentication request is not targeted to the OSP’s whitelisted URLs, OSP rejects the request.

- “Configuring a Whitelist of Target URLs in Identity Applications” on page 16
- “Configuring a Whitelist of Target URLs in OSP” on page 16

Configuring a Whitelist of Target URLs in Identity Applications

You can control which URLs the identity applications can redirect to post logout. This behavior is controlled by `com.novell.pwdmgmt.login.PREF_LOGOUT_WHITELIST` entry in the `ism-configuration.properties` file. To allow identity applications to redirect to a URL after logout, add that URL or a regular expression matching that URL to this entry in one of the following formats:

```
https://google.com
```

or

```
https://www\\.((google)\|(wikipedia))\\.com
```

Configuring a Whitelist of Target URLs in OSP

The whitelist feature is turned on by default. You can manually configure the whitelist entries or disable the whitelist by modifying certain settings in the `ism-configuration.properties` file.

To disable the whitelist, add the following property to the `ism-configuration.properties` file:

```
com.netiq.idm.osp.target-white-list.enabled = false
```

To configure the whitelist manually, add the following property to the `ism-configuration.properties` file:

```
com.netiq.idm.osp.target-white-list.mode = manual
```

Manual mode means that only those URLs or patterns that are configured explicitly in the whitelist are part of the whitelist. You can add one or both of the following properties:

```
com.netiq.idm.osp.target-white-list.uris = <space-separated-list-of-urls>
```

```
com.netiq.idm.osp.target-white-list.uri-patterns = <space-separated-list-
of-url-regex>
```

For example:

com.netiq.idm.osp.target-white-list.uris = `https://www.google.com/ http://bing.com`

com.netiq.idm.osp.target-white-list.uri-patterns = `\\Qhttps\\E:// .*\\Q.provo.novell.com\\E \\Qhttps\\E://.*\\Q.microfocus.com\\E`, where `\\Q` and `\\E` are regex, the regular expression.

Mixed mode, on the other hand, includes URLs and patterns configured automatically by OSP as well as the URLs and patterns that you configure explicitly. To add to the automatically configured whitelist, include `com.netiq.idm.osp.target-white-list.uris` or `com.netiq.idm.osp.target-white-list.uri-patterns` or both properties and specify `com.netiq.idm.osp.target-white-list.mode = mixed`.

Preventing Clickjacking Attacks

If OpenText Identity Manager is deployed in a distributed setup and User Application and OSP are installed on separate servers, the environment can be susceptible to clickjacking attacks. For more information, see HTTP Strict Transport Security (https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security) and Clickjacking (https://en.wikipedia.org/wiki/Clickjacking).

HSTS forces all responses to pass through HTTPS connections instead of plain text HTTP. This ensures that the entire channel is encrypted before any data is sent on the channel and eliminates any chances for the attackers to read or modify the data in transit. To prevent clickjacking attacks, perform the following actions:

Update OSP Server Configuration

1. Stop Tomcat. For example, `systemctl stop netiq-tomcat`
2. Navigate to `<tomcat-install-directory>/conf/web.xml` or `<tomcat-install-directory>\conf\` directory.
3. Add the following filter to the `web.xml` file:

```
<filter>
        <filter-name>httpHeaderSecurity</filter-name>
        <filter-
class>org.apache.catalina.filters.HttpHeaderSecurityFilter</filter-
class>
        <async-supported>true</async-supported>
        <init-param>
             <param-name>antiClickJackingOption</param-name>
           <param-value>ALLOW-FROM</param-value>
   </init-param>
   <init-param>
                    <param-name>antiClickJackingUri</param-name>
                    <param-value>User Application URI</param-value>
   </init-param>
  <init-param>
            <param-name>hstsMaxAgeSeconds</param-name>
            <param-value>31536000</param-value>
        </init-param>
        <init-param>
            <param-name>hstsIncludeSubDomains</param-name>
            <param-value>true</param-value>
        </init-param>
  </filter>

 <filter-mapping>
        <filter-name>httpHeaderSecurity</filter-name>
        <url-pattern>/*</url-pattern>
        <dispatcher>REQUEST</dispatcher>
    </filter-mapping>
```

4. Save the file.
5. Start Tomcat. For example, `systemctl start netiq-tomcat`

For example, User Application URI: https://ua.microfocus.com:8643/, this is where User Application is running.

Update User Application Server Configuration

1 Stop Tomcat. For example, `systemctl stop netiq-tomcat`

2 Navigate to `<tomcat-install-directory>/conf/web.xml` or `<tomcat-install-directory>\conf\` directory.

3 Add the following filter to the `web.xml` file:

```
<filter>
        <filter-name>httpHeaderSecurity</filter-name>
        <filter-
class>org.apache.catalina.filters.HttpHeaderSecurityFilter</filter-
class>
        <async-supported>true</async-supported>
        <init-param>
             <param-name>antiClickJackingOption</param-name>
             <param-value>SAMEORIGIN</param-value>
        </init-param>
        <init-param>
            <param-name>hstsMaxAgeSeconds</param-name>
            <param-value>31536000</param-value>
        </init-param>
        <init-param>
            <param-name>hstsIncludeSubDomains</param-name>
            <param-value>true</param-value>
        </init-param>
  </filter>

 <filter-mapping>
        <filter-name>httpHeaderSecurity</filter-name>
        <url-pattern>/*</url-pattern>
        <dispatcher>REQUEST</dispatcher>
    </filter-mapping>
```

4 Save the file.

5 Start Tomcat. For example, `systemctl start netiq-tomcat.service`

NOTE: As per RFC 7034, the `ALLOW-FROM` parameter supports only a single domain. It does not support multiple domains. For example, if OSP, User Application, SSPR, and Identity Reporting are installed on different computers, this parameter does not work.

Rejecting Client-initiated SSL Renegotiation on Windows

Under certain circumstances, OpenText Identity Manager can be susceptible to a Denial of Service attack caused by a client initiated SSL renegotiation operation. To configure the product to reject this operation, perform the following actions on each computer running the identity applications:

1 Edit the `tomcat-install-directory>\bin\setenv.bat` file.

2 Add the following flag to the `CATALINA_OPTS` entry in the file:

```
"-Djdk.tls.rejectClientInitiatedRenegotiation=true"
```

3 Save the `setenv.bat` file.

4 Restart Tomcat.

Using Tomcat connectionTimeout Attribute to Defend Against Denial-of-Service Attack

If a client sends a request in chunks, the server keeps waiting until the complete request is received. This might be misused to initiate an attack on the server wherein a malicious client overwhelms the server by sending multiple requests at the same time. As the server keeps the connection open waiting for the requests to complete, it causes the server to slowdown. Meanwhile, the server will not be able to respond to the legitimate requests, resulting in a denial-of-service.

You can mitigate this issue by using the `connectionTimeout` attribute. By default, the value of this attribute is set to 60000 (i.e. 60 seconds). If you reduce this value, you can limit the time for which the connection is kept open for each request, and thereby maintain the server in a responsive state, even under denial-of-service attack.

You must perform the following actions to configure the `connectionTimeout` attribute on the server:

1 Stop Tomcat.

2 Navigate to the `conf` directory for Tomcat, located by default in the following directories:

Linux: `/opt/netiq/idm/apps/tomcat/conf/`

Windows: `C:\NetIQ\idm\apps\tomcat\conf`

3 In a text editor, open the `server.xml` file from the `conf` directory.

4 In the connector configuration, add the `connectionTimeout` attribute and set a value, as desired. Below is an extract of connector configuration from the `server.xml` file in which the `connectionTimeout` is set to `20000`.

```
<Connector executor="tomcatThreadPool"
           port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8543" />
```

5 Save the `server.xml` file.

6 Start Tomcat.

Securing Connected Systems

Keep in mind that the connected systems that you are synchronizing data to might store or transport that data in a compromising manner.

Secure the systems with which you exchange passwords. For example LDAP, NIS, and Windows each have security concerns that you must consider before enabling password synchronization with those systems.

Many software vendors provide specific security guidelines that you should follow for their products.

Password Generation

OpenText Identity Manager includes a predefined password generation job for the Job Scheduler. The password generation job generates random passwords for a group of User objects in OpenText eDirectory, either periodically or on demand. This functionality is designed primarily to support products like Certificate Login, but can also be used in other situations.

Invoking the password generation job initializes NMAS with the password policy, and the following occurs for each object in the specified job scope:

1. NMAS generates a random password consistent with the password policy specified in the job. Password policies are stored in `nspmPasswordPolicy` objects. Typically, each connected system has its own policy object. These policy objects can be stored in `DirXML-Driver` and `DirXML-DriverSet` objects.
2. Each generated password is submitted, one at a time, to the containing driver's Subscriber channel.

   If the object has a non-disabled association for the driver then a `<generated-password>` event is submitted to the Subscriber channel event queue (cache) of the driver.

   If the object has no association for the driver and the option to submit events for non-associated objects is selected, then a `<generated-password>` event is submitted to the Subscriber channel event queue (cache) of the driver.
3. It is up to the Subscriber channel policies to handle the generated passwords. The Job Scheduler is only responsible for generating the passwords and handing them off to the Subscriber channel.

Designer for OpenText Identity Manager

When using Designer for OpenText Identity Manager, consider the following points:

- Monitor and control who has rights to create or modify an OpenText Identity Manager driver.

  Administrative rights are needed to create OpenText Identity Manager objects and configure drivers.
- Before giving a consultant an Identity Vault administrator password, limit the rights assigned to that administrator to areas of the tree that the consultant must access.
- Delete the project files (`.proj`) or save them to a company directory.

  Designer `.proj` files are to remain at the company's project site. A consultant does not take the files after completing a project.
- After project files, log files, and trace files are no longer needed, delete them.
- Before discarding or surplusing a laptop, verify that project files have been cleaned.
- Ensure that the connection from Designer to the Identity Vault server is physically secure. Otherwise, someone could monitor the wire and pull sensitive information.
- When you use Document Generator to create documents, be careful with those documents. These documents can contain passwords and sensitive data in clear text.
- If Designer needs to read or write to an OpenText eDirectory attribute, do not mark that attribute as encrypted. Designer is unable to read or write to encrypted attributes.
- Do not store passwords that are sensitive.

Currently, Designer projects are not encrypted. Passwords are only encoded. Therefore, do not share Designer projects that have saved passwords.

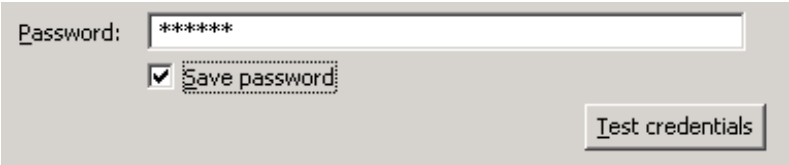
To save a password for a session, but not save it to the project:

1. In an expanded Outline view, right-click an Identity Vault.
2. Select **Properties**.
3. On the Configuration page, type a password, then click **OK**.

   You can enter a password once per session. After you close the project, the password is lost.

To save a password to the hard drive, complete Steps 1-3, select **Save Password**, then click **OK**.

Figure 2-1 *Save Password*



Industry Best Practices for Security

Follow industry best practices for security measures, such as blocking unused ports on the server.

Tracking Changes to Sensitive Information

- "Using Designer to Log Events" on page 21

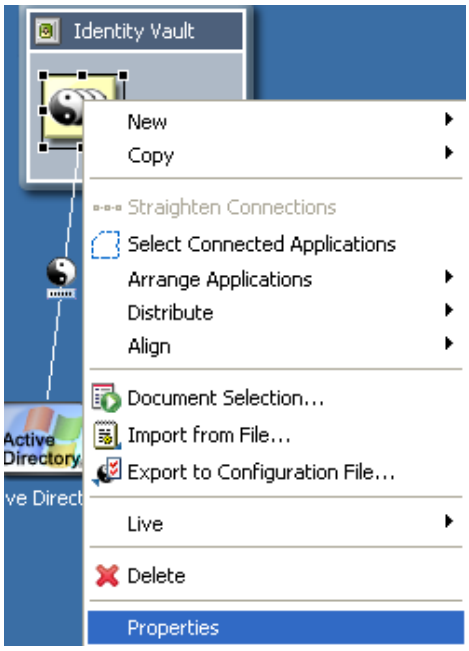
Using Designer to Log Events

You can log events that apply to a driver set or to a driver.

- "Logging Events for a Driver Set" on page 21
- "Logging Events for a Driver" on page 23

Logging Events for a Driver Set

1 In Designer, right-click a driver set, then select **Properties**.



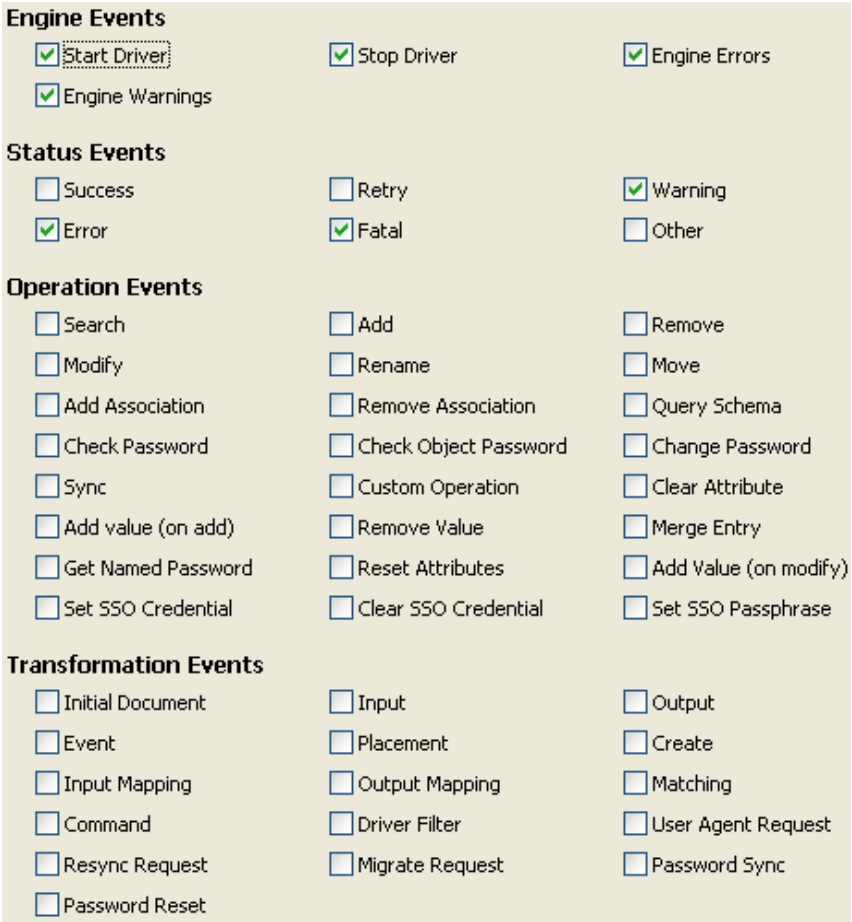
2 Select **Driver Set Log Level**, then select **Log Specific Events**.

3 Click the **Select Events to Log** icon.

4 Enable the **Turn off logging to Driver Set, Subscriber and Publisher logs** option to prevent logging OpenText Identity Manager events to OpenText eDirectory.

   Enabling this option improves the performance of the system.

5 Select events to log, then click **OK**.



Logging Events for a Driver

1 In Designer, right-click a driver, then select **Properties**.

2 Select **Driver Log Level**, then select **Log Specific Events**.

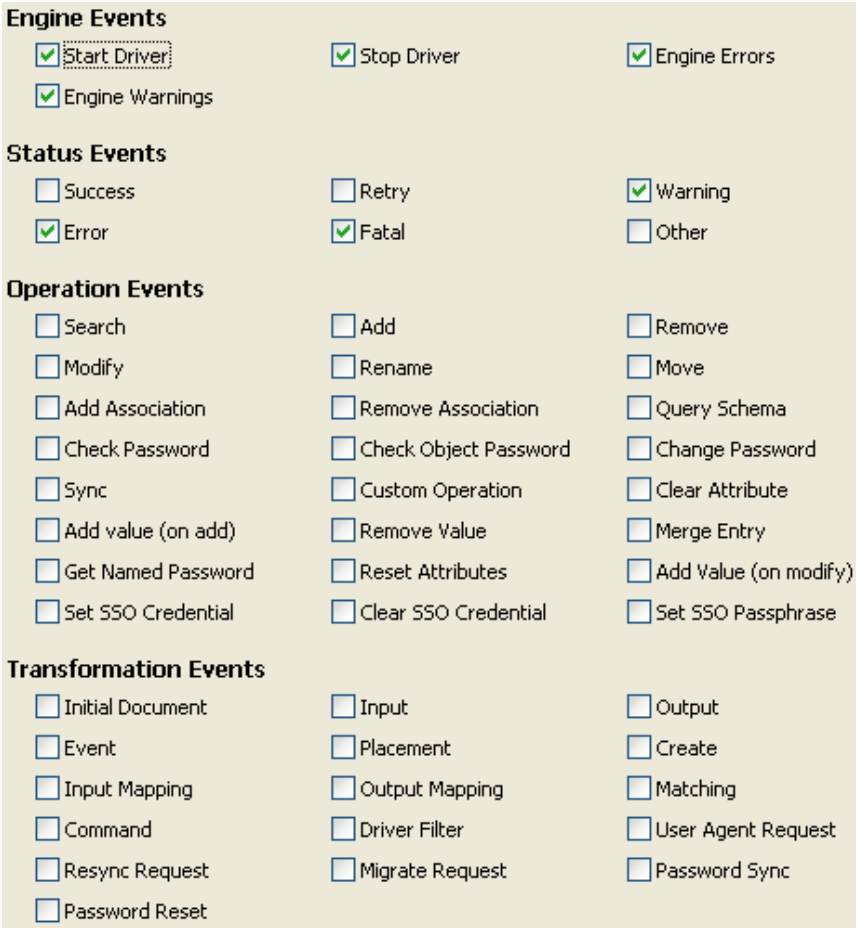
3 If you prefer, you can accept the settings for the driver set, then click **OK**.

or

Deselect **Use log settings from the Driver Set**, select **Log specific events**, then click **OK**.

4 Click the **Select Events to Log** icon.

5 Select events to log, then click **OK**.



Establishing a Security Equivalent User

Security Equivalence refers to an object being equivalent in rights to another object. You can define and deploy security equivalences objects for drivers in the Identity Vault. For example, an Oracle database driver contains a policy to create a user in the Identity Vault in a container every time a user is created in the database, but the driver doesn't have enough permissions on the container to create the user, thus the process fails.

The driver must run with Security Equivalence to a user with sufficient rights. You can set the driver equivalent to an Admin or a similar user. For stronger security, you can define a user with minimal rights necessary for the operations you want the driver to perform.The driver user must be a trustee of the containers where synchronized users and groups reside, with the rights listed in Table 2-2. Inheritance must be set for [Entry Rights] and [All Attribute Rights].

Table 2-2 *Base Container Rights Required by the Driver Security-Equivalent User*

Operation	[Entry Rights]	[All Attribute Rights]
Subscriber notification of account changes (recommended minimum)	Browse	Compare and Read

Operation	[Entry Rights]	[All Attribute Rights]
Creating objects in the Identity Vault without group synchronization	Browse and Create	Compare and Read
Creating objects in the Identity Vault with group synchronization	Browse and Create	Compare, Read, and Write
Modifying objects in the Identity Vault	Browse	Compare, Read, and Write
Renaming objects in the Identity Vault	Browse and Rename	Compare and Read
Deleting objects from the Identity Vault	Browse and Erase	Compare, Read, and Write
Retrieving passwords from the Identity Vault	Browse and Supervisor	Compare and Read
Updating passwords in the Identity Vault	Browse and Supervisor	Compare, Read, and Write

If you do not set Supervisor for [Entry Rights], the driver will not have rights to set passwords. If you do not want to set passwords, you can set the Subscribe setting for the `User class` `nspmDistributionPassword` attribute to `Ignore` in the filter to avoid error messages. For details about accessing and editing the filter, see the appropriate policy publication on the OpenText Identity Manager Documentation Website. For complete information about rights, see "Deploying a Driver to an Identity Vault" in the *OpenText™ Designer for Identity Manager Administration Guide*.

Securing Communication Between ActiveMQ and Identity Applications

OpenText recommends using Secure Socket Layer (SSL) protocols for communication between Identity Applications and ActiveMQ. By default, the SSL protocol is not configured between these two components. You must configure the SSL connection between them:

- Between ActiveMQ and Identity Applications

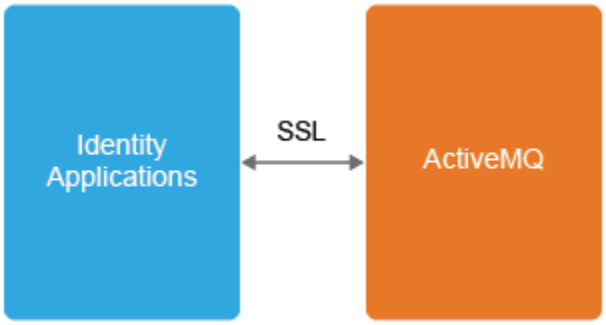
  Refer to the following sections for instructions:

  - "Creating a Keystore and a Truststore" on page 26
  - "Enabling SSL for ActiveMQ" on page 27
  - "Enabling SSL for the Identity Applications Client" on page 27
- Between Identity Applications and ActiveMQ

  Refer to the following sections for instructions:

  - "Creating a Keystore and a Truststore" on page 26
  - "Enabling SSL for ActiveMQ" on page 27
  - "Enabling SSL for the Identity Applications Client" on page 27

Figure 2-2 Secured Connection between ActiveMQ and Identity Applications



To support the SSL connections, you need to create keystore and truststore files. This section explains how to create, export, and store this certificate on your server

After the secured connection is enabled, the Identity Applications perform an SSL handshake to establish a secure channel.

Creating a Keystore and a Truststore

1. Create a certificate for the broker by using the keytool.

   ```
   keytool -genkey -alias broker -keyalg RSA -keystore broker.ks
   ```

2. Export the broker's certificate to share with clients.

   ```
   keytool -export -alias broker -keystore broker.ks -file broker_cert
   ```

3. Create a certificate/keystore for the client.

   ```
   keytool -genkey -alias client -keyalg RSA -keystore client.ks
   ```

4. Create a truststore for the client and import the broker's certificate. This establishes that the client trusts the broker.

   ```
   keytool -import -alias broker -keystore client.ts -file broker_cert
   ```

5. Export the client's certificate so it can be shared with broker:

   ```
   keytool -export -alias client -keystore client.ks -file client_cert
   ```

6. Create a truststore for the broker, and import the client's certificate. This establishes that the broker trusts the client:

   ```
   keytool -import -alias client -keystore broker.ts -file client_cert
   ```

NOTE: You must use the same passwords that were used for creating the keystores to configure the Fanout components for SSL. For more information about creating certificates, see Setting up the Key and Trust Stores.

Enabling SSL for ActiveMQ

1 Copy the files `broker.ks` and `broker.ts` to the server where ActiveMQ is installed.

2 Edit the `activemq.xml` file.

   2a Navigate to the `<AMQ install path>/conf/` folder and open the `activemq.xml` file.

   2b Add the transport connector to enable SSL in the transportConnectors section.

```
<transportConnector name="ssl" uri="ssl://
0.0.0.0:61617?trace=true"/>
```

   2c Add the keystore or truststore paths and their passwords in the sslContext section.

```
<sslContext> <sslContext keyStore="file:///root/activemqkeystore/
broker.ks" keyStorePassword="novell" trustStore="file:///root/
activemqkeystore/broker.ts" trustStorePassword="novell"/> </
sslContext>
```

3 Restart ActiveMQ.

NOTE: For more information about enabling secure transport on ActiveMQ, see Using Spring to configure SSL for a Broker instance.

Enabling SSL for the Identity Applications Client

1 Go to `tomcat/conf` folder and modify the resource tag containing jms connection property with below values in the `server.xml` file:

```
<Resource auth="Container" brokerName="LocalActiveMQBroker"
brokerURL="ssl://0.0.0.0:61617" description="JMS Connection Factory"
factory="org.apache.activemq.jndi.JNDIReferenceFactory" name="jms/
ConnectionFactory"
type="org.apache.activemq.ActiveMQSslConnectionFactory"
trustStore="file:///root/activemqkeystore/client.ts"
trustStorePassword="novell" keyStore="file:///root/activemqkeystore/
client.ks" keyStorePassword="novell"/>
```

NOTE: Update the path of the truststore and keystore files accordingly.

---

2 Restart the Identity Applications server.

Securing Cloud Deployment of OpenText Identity Manager Components

You can deploy the components on MicroSoft Azure and Amazon Web Services EC2. While deploying the components on these cloud platforms, you must consider the following security recommendations:

- The components should be configured on a private network with no public access
- Web applications such as Identity Applications, Identity Reporting, or OpenText Identity Console should be accessed through an application gateway

- The components should be configured to use a secured communication channel
- The following ports should be made available on the OpenText Identity Manager servers to use within the subnet for MS Azure:

Component	Port	Description
LDAP for Identity Vault	TCP 636	Required for the secured LDAP communication.
Identity Applications	TCP 8543	Required for the HTTPS communication to access Identity Applications.
Identity Reporting	TCP 8643	Required for the HTTPS communication to access Identity Reporting.

- The following ports should be made available on the OpenText Identity Manager servers to use within the subnet for AWS EC2:

Component	Port	Description
LDAP for Identity Vault	TCP 636	Required for the secured LDAP communication.
Identity Applications	TCP 8543	Required for the HTTPS communication to access Identity Applications.
Identity Reporting	TCP 8643	Required for the HTTPS communication to access Identity Reporting.
PostgreSQL Database	TCP 5432	Required for the secured database communication to access PostgreSQL.

Securing the Information with is-sensitive Attribute

The DirXML Script and XSLT methods of creating policies apply `is-sensitive` attribute on the XDS nodes to hide values of sensitive attributes such as passwords in the trace file. When this attribute is used with a driver, OpenText Identity Manager prints the trace output similar to the following:

```
Driver : Applying policy: %+C%14Cis-sensetive%-C.
Driver : Applying to add #1.
Driver :    Evaluating selection criteria for rule 'is-senesitive '.
Driver :      (if-attr 'secret' available) = TRUE.
Driver :    Rule selected.
Driver :    Applying rule 'is-senesitive '.
Driver :      Action: do-set-xml-attr("is-sensitive","*[@attr-
name='secret']","true").
Driver :        arg-string("true")
Driver :          token-text("true")
Driver :          Arg Value: "true".
Driver :Policy returned:
Driver :
<nds dtdversion="4.0" ndsversion="8.x">
  <source>
    <product version="4.6.0.0">DirXML</product>
    <contact>NetIQ Corporation</contact>
  </source>
  <input>
    <add class-name="User" src-dn="data/users/user1">
      <add-attr attr-name="secret" is-sensitive="true"><!-- content
suppressed -->
      </add-attr>
    </add>
  </input>
</nds>
```

Configuring Stronger Ciphers for SSL Communication

You can configure OpenText Identity Manager in Suite B mode to enhance the security requirements of the environment.

Suite B requirement originated from the National Security Agency (NSA) to specify a cryptographic interoperability strategy. Suite B includes the following cryptographic algorithms:

- Encryption based on the Advanced Encryption Standard (AES) using 128-bit keys or 256-bit keys
- Digital signatures with the Elliptic Curve Digital Signature Algorithm (ECDSA) on P-256 and P-384 curves
- Key exchange, either pre-shared or dynamic, using the Elliptic Curve Diffie-Hellman (ECDH) method on P-256 and P-384 curves
- Hashing (digital fingerprinting) based on the Secure Hash Algorithm-2 (SHA-256 and SHA-384)

NOTE: Suite B standard is subject to change. NSA may change their recommendations in future. Suite B support in OpenText Identity Manager is based on our interpretation of the NSA recommendations. For more information about Suite B, see Suite B Cryptography.

Prerequisites

To configure OpenText Identity Manager in Suite B mode, your environment must meet the following conditions:

- OpenText eDirectory 9.0.2 or later is installed as an Identity Vault
- TLS 1.2 is enforced as a communication protocol
- Suite B connection parameter is specified in the driver, Remote Loader, or Fan-Out configuration to enforce the Suite B specification for a secured communication

NOTE: In Suite B mode, the SSL connection is restricted to accept only Suite B supported certificates. If a certificate is expired or invalid, the handshake fails and the communication is not established. For generating Suite B certificates, see "Creating a Server Certificate Object" in the *OpenText eDirectory Administration Guide*.

The following table lists the requirements as specified by Suite B:

Requirement	Description
Protocol	TLS 1.2 is supported in Suite B mode.
Public keys	The public key for certificates must be a minimum size of EC 256 bits.
Signature algorithm	The signature algorithm for certificates must be a minimum size of ECDSA 256 bits (curve P256) and SHA256.
Hash algorithm	The hash algorithm must have the minimum size of SHA256.
Cipher specification	The following ciphers are supported for Suite B mode: ◆ TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 ◆ TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 To use ciphers with stronger signature and hash algorithms, the certificates of server key file must contain similar or stronger signature and hash algorithms. Suite B supports two levels of cryptographic security: 128 bit and 192 bit. The level defines a minimum strength that all cryptographic algorithms must provide. In Suite B 192-bit processing mode, the supported cipher suite is TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384.

Configuring the Settings for Suite B Mode

To meet the requirements specified by Suite B, you must specify the appropriate settings in the OpenText Identity Manager components. This section provides information about those settings.

Engine

To configure the OpenText Identity Manager engine in Suite B mode, you must set the Suite B configuration option **enforceSuiteB** = **true** in the driver configuration by using Designer.

With the use of stronger ciphers in Suite B mode, passwords managed by the engine such as named password, application password, and Remote Loader password will be re-encrypted when they are used for the first time after upgrading the engine to 4.6 version. On an upgraded engine, the existing encrypted attribute values in the driver cache file are not re-encrypted with stronger ciphers because they are removed from the TAO file when the event is processed. However, when new encrypted attributes are stored in the cache, they are encrypted with AES 256-bit keys.

Engine and Remote Loader Communication

To make the engine and Remote Loader communication compliant with Suite B mode, set the Suite B configuration option **enforceSuiteB** = **true** in the driver configuration. The Suite B communication can also be configured in the Remote Loader configuration file for a driver by setting **enforceSuiteB** to `true`. For more information, see Configuring the Remote Loader and Drivers in the *OpenText™ Identity Manager CE 24.4 (v4.10) Driver Administration Guide*.

Suite B mode is disabled by default. When you enable it, OpenText Identity Manager automatically uses TLS 1.2 or later for communication. If you try to connect a Suite B-enabled engine with a Remote Loader that does not support TLSv1.2, the handshake fails and the communication is not established. For example, Remote Loader 4.5.3, which does not support TLS v1.2.

Engine and Fan-Out Agent Communication

For enabling the Suite B communication, manually include `netiq.fanoutagent.connection.enforceSuiteB=true` parameter in the Fan-Out Agent configuration file. You also need to specify **enforceSuiteB = true** in the driver configuration. Suite B configuration is supported with driver version 1.0.1.1. For more information, see the OpenText™ Identity Manager Driver for JDBC Fanout Implementation Guide (https://www.netiq.com/documentation/identity-manager-4.10-drivers/jdbc_fanout/data/netiq-identity-manager-for-jdbc-fan-out-driver-implementation-guide.html)

OpenText Identity Manager Drivers

Along with the configuration changes discussed in the earlier sections, additional changes have been made to these drivers to enable them for Suite B.

OpenText eDirectory to OpenText eDirectory Driver

The eDir-to-eDir Driver Certificates Wizard in Designer allows the use of stronger ciphers for encrypting the data as specified by Suite B. You import the Suite B compliant certificates into the certificate store that the driver uses. For more information, see Securing Driver Communication in the *OpenText™ Identity Manager Driver for eDirectory Implementation Guide (https://www.netiq.com/documentation/identity-manager-4.10-drivers/edirectory/data/bookinfo.html)*.

Active Directory Driver

The driver stores the password in the Windows registry. For Suite B compliance, the driver uses AES 256-bit encryption algorithm to encrypt the new passwords.

Passwords that are already in the registry are not re-encrypted with stronger ciphers because they are cleaned up when the event is processed. However, when new passwords are stored in the registry, they are encrypted with AES 256-bit keys.

Enabling Stronger Ciphers for SSL Communication

By default, OpenText Identity Manager supports the 128-bit SSL communication between the engine and the Remote Loader/ Fan-Out agent. The supported ciphers include:

- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

Oracle provides a default cryptographic jurisdiction policy file that limits the strength of cryptographic algorithms. When using stronger ciphers, you must increase the strength of encryption used. Cipher suites using key lengths greater than 128 bits, such as 256-bit AES encryption, require the JCE Unlimited Strength Jurisdiction policy files that enable additional cipher suites for Java in a separate JAR file.

To enable 256-bit or higher ciphers:

1. Download and extract the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files zipped file from Oracle's Java website to a temporary folder on your computer.

   For example, download Java 8 JCE files from Oracle’s download page.
2. Navigate to the JRE path of your product installation directory and save the `local.policy.jar` and `US_export_policy.jar` files to a different directory.

   For example: `/opt/novell/eDirectory/lib64/nds-modules/jre/lib/security`
3. Replace these policy jars with the files you extracted in Step 1.

   For detailed instructions, see the steps listed in the `Readme.txt` file included in the zipped file.

Verifying the Suite B Settings

When Suite B mode is enabled, the trace shows the cipher values and the TLS version that is used in the SSL communication.

To verify whether the communication is successful in Suite B mode, include a non-EC certificate in the configuration and verify that the trace file messages indicate that the Suite B certificate is not imported. For example, when Suite B is successfully enabled, the trace file can show entries similar to this:

```
[11/28/16 16:34:14.828]:Delimited Text PT:JSSEKMO: [KMO2Keystore:readKMOFromIDV] - Created jclient context : Context=8847472, epoch=1
[11/28/16 16:34:14.829]:Delimited Text PT:JSSEKMO: [KMO2Keystore:readKMOFromIDV] - Id resolution successful. Entry Id : 34269
[11/28/16 16:34:14.829]:Delimited Text PT:JSSEKMO: [KMO2Keystore:readKMOFromIDV] - Successfully read the KMO attributes from IDV
[11/28/16 16:34:14.829]:Delimited Text PT:JSSEKMO: [KMO2Keystore:readKMOFromIDV] - Initialized public key bytes..
[11/28/16 16:34:14.829]:Delimited Text PT:JSSEKMO: [KMO2Keystore:readKMOFromIDV] - Initialized private key bytes..
[11/28/16 16:34:14.829]:Delimited Text PT:JSSEKMO: [KMO2Keystore:readKMOFromIDV] - Freeing up the context.
[11/28/16 16:34:14.830]:Delimited Text PT:JSSEKMO: [KMO2Keystore:parseCertificate] - Certificate parsed successfully...
[11/28/16 16:34:14.830]:Delimited Text PT:JSSEKMO: [KMO2Keystore:initPrivateKey] - Successfully unwrapped the secret.
[11/28/16 16:34:14.830]:Delimited Text PT:JSSEKMO: [KMO2Keystore:initPrivateKey] - cleaning up context.
[11/28/16 16:34:14.831]:Delimited Text PT:JSSEKMO: [KMO2Keystore:setupKeystore] - Initialized the keystore.
[11/28/16 16:34:14.831]:Delimited Text PT:JSSEKMO: [KMO2Keystore:setupKeystore] - keystore load completed.
[11/28/16 16:34:14.831]:Delimited Text PT:JSSEKMO: [KMO2Keystore:setupKeystore] - Instantiated certificate factory.
[11/28/16 16:34:14.832]:Delimited Text PT:JSSEKMO: [KMO2Keystore:setupKeystore] - Successfully setup the certificate chain.
[11/28/16 16:34:14.832]:Delimited Text PT:JSSEKMO: [KMO2Keystore:initKsWithKeyPair] - Successfully generated private key from bytes.
[11/28/16 16:34:14.839]:Delimited Text PT:JSSEKMO: [KMO2Keystore:initKsWithKeyPair] - Successfully created a keystore entry.
[11/28/16 16:34:14.840]:Delimited Text PT:Remote Interface Driver: Creating an JSSEKmoFactory ServerSocket
[11/28/16 16:34:15.109]:Delimited Text PT:Remote Interface Driver: Cipher Suite : TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
[11/28/16 16:34:15.110]:Delimited Text PT:Remote Interface Driver: JSSE Socket, cipher suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 , peer host:
net.ssl.SSLPeerUnverifiedException: peer not authenticated
[11/28/16 16:34:15.110]:Delimited Text PT:Remote Interface Driver: Connection established...
```

In case of an error, you will see messages similar to this:

```
DirXML Log Event -------------------
     Driver:   \SLES17885\system\Driver Set\Delimited Text Driver
     Channel:  Publisher
     Status:   Error
     Message:  java.io.IOException: Suite B certificate is not imported
        at com.novell.nds.dirxml.remote.SocketStream.connect(SocketStream.java:629)
        at com.novell.nds.dirxml.remote.Connection.connectStream(Connection.java:710)
        at com.novell.nds.dirxml.remote.Connection.connect(Connection.java:386)
        at com.novell.nds.dirxml.remote.driver.PublicationShimImpl.start(PublicationShimImpl.java:113)
        at com.novell.nds.dirxml.engine.Publisher.run(Publisher.java:607)
        at java.lang.Thread.run(Thread.java:745)
```

If the certificate is not valid, the trace shows messages similar to this:

```
[02/23/17 19:46:52.294]:Delimited Text Driver PT:JSSEKMO: [KMO2Keystore:parseCertificate] - Certificate parsed successfully...
[02/23/17 19:46:52.302]:Delimited Text Driver PT:JSSEKMO: [KMO2Keystore:initPrivateKey] - Successfully unwrapped the secret.
[02/23/17 19:46:52.303]:Delimited Text Driver PT:JSSEKMO: [KMO2Keystore:initPrivateKey] - cleaning up context.
[02/23/17 19:46:52.303]:Delimited Text Driver PT:JSSEKMO: [KMO2Keystore:setupKeystore] - Initialized the keystore.
[02/23/17 19:46:52.310]:Delimited Text Driver PT:JSSEKMO: [KMO2Keystore:setupKeystore] - keystore load completed.
[02/23/17 19:46:52.314]:Delimited Text Driver PT:JSSEKMO: [KMO2Keystore:setupKeystore] - Instantiated certificate factory.
[02/23/17 19:46:52.316]:Delimited Text Driver PT:JSSEKMO: [KMO2Keystore:setupKeystore] - Successfully setup the certificate chain.
[02/23/17 19:46:52.317]:Delimited Text Driver PT:JSSEKMO: [KMO2Keystore:initKsWithKeyPair] - Successfully generated private key from bytes.
[02/23/17 19:46:52.322]:Delimited Text Driver PT:JSSEKMO: [KMO2Keystore:initKsWithKeyPair] - Successfully created a keystore entry.
[02/23/17 19:46:52.333]:Delimited Text Driver PT:Remote Interface Driver: Creating an JSSEKmoFactory ServerSocket
[02/23/17 19:46:52.393]:Delimited Text Driver PT:Receiving DOM document from application.
[02/23/17 19:46:52.394]:Delimited Text Driver PT:
<nds dtdversion="4.0" ndsversion="8.x">
  <input>
    <status level="error" type="remoteloader">java.io.IOException: Error during SSL handshake
        at com.novell.nds.dirxml.remote.SocketStream.connect(SocketStream.java:619)
        at com.novell.nds.dirxml.remote.Connection.connectStream(Connection.java:710)
        at com.novell.nds.dirxml.remote.Connection.connect(Connection.java:386)
        at com.novell.nds.dirxml.remote.driver.PublicationShimImpl.start(PublicationShimImpl.java:113)
        at com.novell.nds.dirxml.engine.Publisher.run(Publisher.java:607)
        at java.lang.Thread.run(Thread.java:745)
</status>
  </input>
</nds>
```