Tivoli® Directory Integrator

IBM

for version 6.1.1

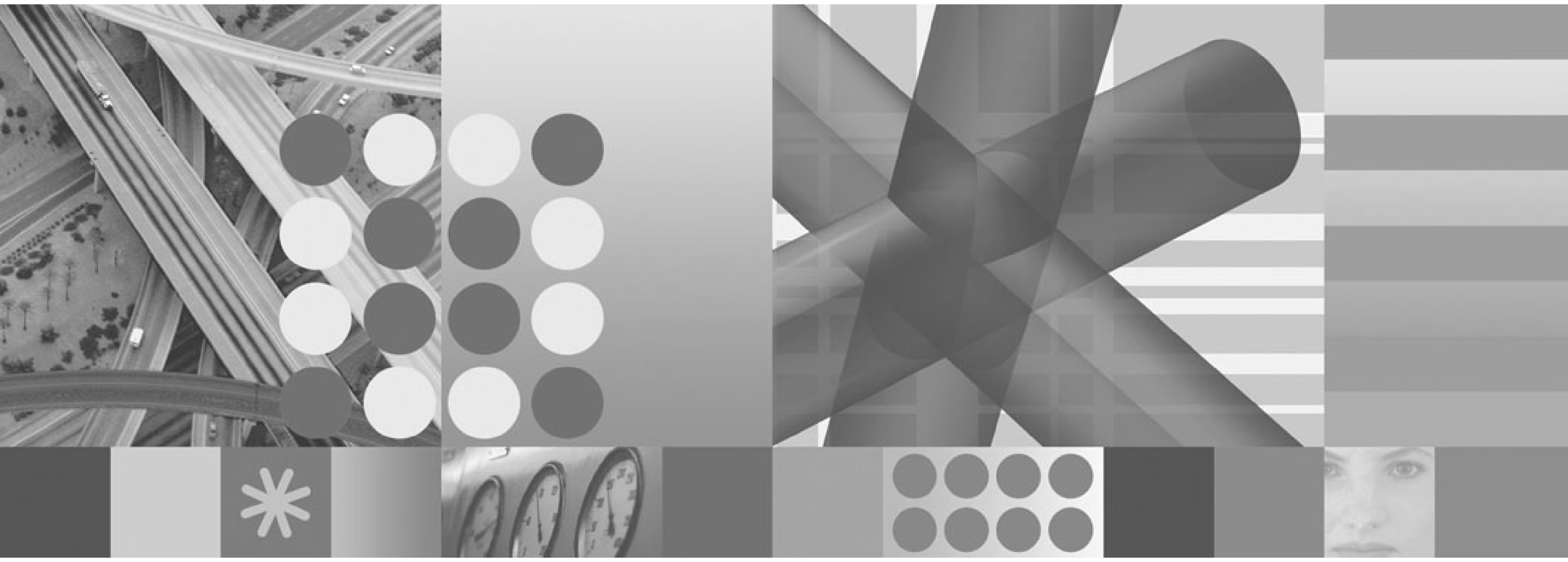
# Password Synchronization Plug-ins Guide

SC32-2563-01

Tivoli® Directory Integrator

IBM

for version 6.1.1



# Password Synchronization Plug-ins Guide

SC32-2563-01

**Note**

**Note:** Before using this information and the product it supports, read the general information under Appendix D, "Notices," on page 123.

**Second Edition (February 2007)**

This edition applies to version 6.1.1 of the IBM Tivoli Directory Integrator and to all subsequent releases and modifications until otherwise indicated in new editions.

# Contents

# Preface

This document describes the procedural steps that are required to achieve password synchronization between IBM® Tivoli® Directory Integrator and a number of IBM and third party products.

The chapters in this guide cover all the plug-ins available for password synchronization at the time of publication. Please see the IBM Tivoli Directory Integrator Web site for any later plug-ins or updates.

This document assumes that both Tivoli Directory Integrator and the products to be integrated are already installed, configured and running on your network. No details are provided regarding the installation and configuration of these products, except where necessary to achieve integration.

## Who should read this book

Tivoli Directory Integrator components are designed for network administrators who are responsible for maintaining user directories and other resources.

This document assumes that you have practical experience installing and using both IBM Tivoli Directory Integrator and the product to be integrated.

## Publications

Read the descriptions of the IBM Tivoli Directory Integrator library and the related publications to determine which publications you might find helpful. After you determine the publications you need, refer to the instructions for accessing publications online.

### IBM Tivoli Directory Integrator library

The publications in the IBM Tivoli Directory Integrator library are:

*IBM Tivoli Directory Integrator 6.1.1: Getting Started*
A brief tutorial and introduction to IBM Tivoli Directory Integrator 6.1.1.

*IBM Tivoli Directory Integrator 6.1.1: Administrator Guide*
Includes complete information for installing the IBM Tivoli Directory Integrator. Includes information about migrating from a previous version of IBM Tivoli Directory Integrator. Includes information about configuring the logging functionality of IBM Tivoli Directory Integrator. Also includes information about the security model underlying the Remote Server API.

*IBM Tivoli Directory Integrator 6.1.1: Users Guide*
Contains information about using the IBM Tivoli Directory Integrator 6.1.1 tool. Contains instructions for designing solutions using the IBM Tivoli Directory Integrator tool (**ibmditk**) or running the ready-made solutions from the command line (**ibmdisrv**). Also provides information about interfaces, concepts and AssemblyLine/EventHandler creation and management. Includes examples to create interaction and hands-on learning of IBM Tivoli Directory Integrator 6.1.1.

*IBM Tivoli Directory Integrator 6.1.1: Reference Guide*
Contains detailed information about the individual components of IBM Tivoli Directory Integrator 6.1.1 AssemblyLine (Connectors, EventHandlers, Parsers, Plug-ins, and so forth).

*IBM Tivoli Directory Integrator 6.1.1: Problem Determination Guide*
Provides information about IBM Tivoli Directory Integrator 6.1.1 tools, resources, and techniques that can aid in the identification and resolution of problems.

*IBM Tivoli Directory Integrator 6.1.1: Messages Guide*
Provides a list of all informational, warning and error messages associated with the IBM Tivoli Directory Integrator 6.1.1.

*IBM Tivoli Directory Integrator 6.1.1: Password Synchronization Plug-ins Guide*
Includes complete information for installing and configuring each of the five IBM Password Synchronization Plug-ins: Windows Password Synchronizer, Sun ONE Directory Server Password Synchronizer, IBM Directory Server Password Synchronizer, Domino Password Synchronizer and Password Synchronizer for UNIX® and Linux®. Also provides configuration instructions for the LDAP Password Store and MQe Password Store.

*IBM Tivoli Directory Integrator 6.1.1: Release Notes*
Describes new features and late-breaking information about IBM Tivoli Directory Integrator 6.1.1 that did not get included in the documentation.

## Related publications

Information related to the IBM Tivoli Directory Integrator is available in the following publications:

- IBM Tivoli Directory Integrator 6.1.1 uses the JNDI client from Sun Microsystems. For information about the JNDI client, refer to the *Java™ Naming and Directory Interface™ 1.2.1 Specification* on the Sun Microsystems Web site at http://java.sun.com/products/jndi/1.2/javadoc/index.html.
- The Tivoli Software Library provides a variety of Tivoli publications such as white papers, datasheets, demonstrations, redbooks, and announcement letters. The Tivoli Software Library is available on the Web at: http://www.ibm.com/software/tivoli/library/
- The *Tivoli Software Glossary* includes definitions for many of the technical terms related to Tivoli software. The *Tivoli Software Glossary* is available on the World-Wide Web, in English only, athttp://publib.boulder.ibm.com/tividd/glossary/tivoliglossarymst.htm

## Accessing publications online

The publications for this product are available online in Portable Document Format (PDF) or Hypertext Markup Language (HTML) format, or both in the Tivoli software library: http://www.ibm.com/software/tivoli/library.

To locate product publications in the library, click the **Product manuals** link on the left side of the Library page. Then, locate and click the name of the product on the Tivoli software information center page.

Information is organized by product and includes READMEs, installation guides, user's guides, administrator's guides, and developer's references as necessary.

**Note:** To ensure proper printing of PDF publications, select the **Fit to page** check box in the Adobe Acrobat Print window (which is available when you click **File->Print**).

## Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use information technology products successfully. With TDI 6.1.1, you can use assistive technologies to hear and navigate the interface. After installation you also can use the keyboard instead of the mouse to operate all features of the graphical user interface.

### Accessibility features

The following list includes the major accessibility features in TDI 6.1.1:

- Supports keyboard-only operation.
- Supports interfaces commonly used by screen readers.
- Discerns keys as tactually separate, and does not activate keys just by touching them.
- Avoids the use of color as the only way to communicate status and information.
- Provides accessible documentation.

### Keyboard navigation

This product uses standard Microsoft®Windows® navigation keys for common Windows actions such as access to the File menu, copy, paste, and delete. Actions that are unique to TDI use TDI keyboard shortcuts. Keyboard shortcuts have been provided wherever needed for all actions.

### Interface Information

The following points include accessibility features of the TDI 6.1.1 user interface and documentation:

- Steps for changing fonts, colors, and contrast settings in the Config Editor (CE):
  1. Type **Alt-F** to access the CE **File** menu. Using the downward arrow, select **Edit Preferences** and press **Enter**.
  2. Under the **Appearance** tab, select **Theme** settings to change the font.
  3. Under **Theme Colors**, select the colors for the CE, and by selecting colors, you can also change the contrast.
- The TDI 6.1.1 Information Center and its related publications are accessibility-enabled for the JAWS screen reader and the IBM Home Page Reader. You can operate all documentation features using the keyboard instead of the mouse.

### Vendor software

The IBM Tivoli Directory Integrator installer uses the FLEXnet Publisher Installation Module (FNPIM).

### Related accessibility information

Visit the *IBM Accessibility Center* at http://www.ibm.com/able for more information about IBM's commitment to accessibility.

## Contacting Software Support

Contact IBM Software Support by using the methods described in the *IBM Software Support Guide* at the following Web site:

http://techsupport.services.ibm.com/guides/handbook.html

The guide provides the following information:

- Registration and eligibility requirements for receiving support
- Telephone numbers, depending on the country in which you are located
- A list of information you should gather before contacting customer support

For more information, see Appendix C, "IBM Software Support," on page 119.

## Tivoli technical training

For Tivoli technical training information, refer to the IBM Tivoli Education Web site: http://www.ibm.com/software/tivoli/education.

## Conventions Used in this Book

The following typeface conventions are used in this book:

**Bold** Lowercase commands or mixed case commands that are difficult to distinguish from surrounding text, keywords, parameters, options, names of Java® classes, and objects are in **bold**.

*Italic* Variables, titles of publications, and special words or phrases that are emphasized are in *italic*.

`Monospace`
Code examples, command lines, screen output, file and directory names that are difficult to distinguish from surrounding text, system messages, text that the user must type, and values for arguments or command options are in `monospace`.

# Chapter 1. Introducing Password Synchronization Plug-ins

This chapter introduces the range of password synchronization plug-ins for IBM Tivoli Directory Integrator . It describes standard concepts, components and procedures.

This chapter contains the following sections:

## Overview

The IBM Tivoli Directory Integrator provides an infrastructure and a number of ready-to-use components for implementing solutions that synchronize user passwords in heterogeneous software environments.

A password synchronization solution built with the IBM Tivoli Directory Integrator can intercept password changes on a number of systems. The intercepted changes can be directed back into:
- The same software systems, or
- A different set of software systems.

Synchronization is achieved through the IBM Tivoli Directory Integrator AssemblyLines, which can be configured to propagate the intercepted passwords to desired systems.

## Building blocks

The components that make up a password synchronization solution are:

**Password Synchronizers**
Components which are deployed on the system where password changes occur. They are responsible for intercepting plain (unencrypted) values of the passwords as they are changed.

**Password Stores**
Components that receive the intercepted passwords, encrypt and store them in locations that can be accessed by the IBM Tivoli Directory Integrator .

**Connectors**
These are either standard or specialized IBM Tivoli Directory Integrator Connectors. They connect to locations where the intercepted and encrypted passwords are stored and are able to retrieve and decrypt the passwords.

**AssemblyLines**
The AssemblyLines use Connectors to get the intercepted passwords and then build custom logic for sending the passwords to other software systems.

**EventHandlers (optional)**
The use of EventHandlers can further automate or schedule the password synchronization process.

## Building the solution

The Password Synchronizers, Password Stores and Connectors are ready-to-use components included in the IBM Tivoli Directory Integrator . As a result, implementing the solution that intercepts the passwords and makes them accessible from IBM Tivoli Directory Integrator is achieved by deploying and configuring these components.

For the part of the solution that consolidates passwords intercepted from different sources and feeds these passwords into systems that need to be synchronized, a custom AssemblyLine must be implemented. The look of the AssemblyLine depends mostly on the custom environment and the requirements for the particular solution. IBM Tivoli Directory Integrator does not include these AssemblyLines; they are implemented by the customer.

A password synchronization AssemblyLine usually uses Iterator Connectors to retrieve passwords from the Password Stores. The AssemblyLine then uses other standard Connectors to set these passwords into other systems. If the systems that are synchronized have custom requirements for setting user passwords, these requirements must be addressed in the AssemblyLine and the Connectors that set these passwords. Such customization might consist of setting certain Connector parameters, for example, turning on the **Auto Map AD Password** option in the LDAP Connector to set user passwords in Active Directory. In more complex cases, scripting might be necessary.

A password synchronization solution might include IBM Tivoli Directory Integrator EventHandlers to automate the process of synchronization. For example, an EventHandler might listen for changes in the repository where a Password Store component stores the intercepted passwords and trigger the synchronization AssemblyLine whenever a new password is intercepted. Another example might be using a Timer EventHandler that starts the synchronization AssemblyLine on a schedule.

Represented here are some basic and common steps. Each of the components mentioned previously provides interfaces which facilitate the tuning of behavior. Also, the various components can be combined with each other to create custom solutions. These key features provide flexibility for building solutions that meet custom requirements and limitations. The password synchronization suite is mostly comprised of the specialized components that intercept the passwords and make them accessible for the IBM Tivoli Directory Integrator . Once the IBM Tivoli Directory Integrator can access the intercepted passwords through its Connectors, the whole flexibility and openness of the IBM Tivoli Directory Integrator architecture can be leveraged in organizing the process of password retrieval and propagation to other systems.

## Available specialized components

The following sections describe the specialized password synchronization components that are currently available.

## Password Synchronizers

**Password Synchronizer for Windows 2000/2003/XP**
Intercepts the Windows login password change. See Chapter 3, “Windows Password Synchronizer,” on page 15.

**Password Synchronizer for IBM Tivoli Directory Server**
Intercepts IBM Tivoli Directory Server password changes. See Chapter 4, “Sun ONE Directory Server Password Synchronizer,” on page 25.

**Password Synchronizer for Sun ONE Directory Server**
Intercepts Sun ONE Directory Server password changes. See Chapter 5, “IBM Directory Server Password Synchronizer,” on page 35.

**Password Synchronizer for Domino®**
Intercepts changes of the HTTP password for Lotus® Notes® users. See Chapter 6, “Domino HTTP Password Synchronizer,” on page 45.

**Password Synchronizer for UNIX and Linux**
Intercepts changes of UNIX and Linux user passwords where PAM is enabled. See Chapter 7, “Password Synchronizer for UNIX and Linux,” on page 63.

## Password Stores

**LDAP Password Store**
Provides the function necessary to store the intercepted user passwords in LDAP directory servers. See Chapter 8, “LDAP Password Store,” on page 73.

**MQ Everyplace® Password Store**
Provides the function necessary to store user passwords into IBM WebSphere® MQ Everyplace. See Chapter 9, “MQ Everyplace Password Store,” on page 85.

## Specialized Connectors

**MQe Password Store Connector**
Provides the function necessary to retrieve password update messages from IBM WebSphere MQ Everyplace and send them to IBM Tivoli Directory Integrator. See Chapter 9, “MQ Everyplace Password Store,” on page 85.

**Tivoli Identity Manager Integration**

This guide also details the steps required for integration between Tivoli Identity Manager and the following Password Synchronizers:

- Sun ONE Directory Server Password Synchronizer,
- IBM Directory Server Password Synchronizer,
- Domino HTTP Password Synchronizer, and
- Password Synchronizer for UNIX and Linux.

See Chapter 10, “Tivoli Identity Manager Integration,” on page 111.

## Password Synchronization Architecture and Workflow

There are several layers in the IBM Tivoli Directory Integrator Password Synchronizer architecture.

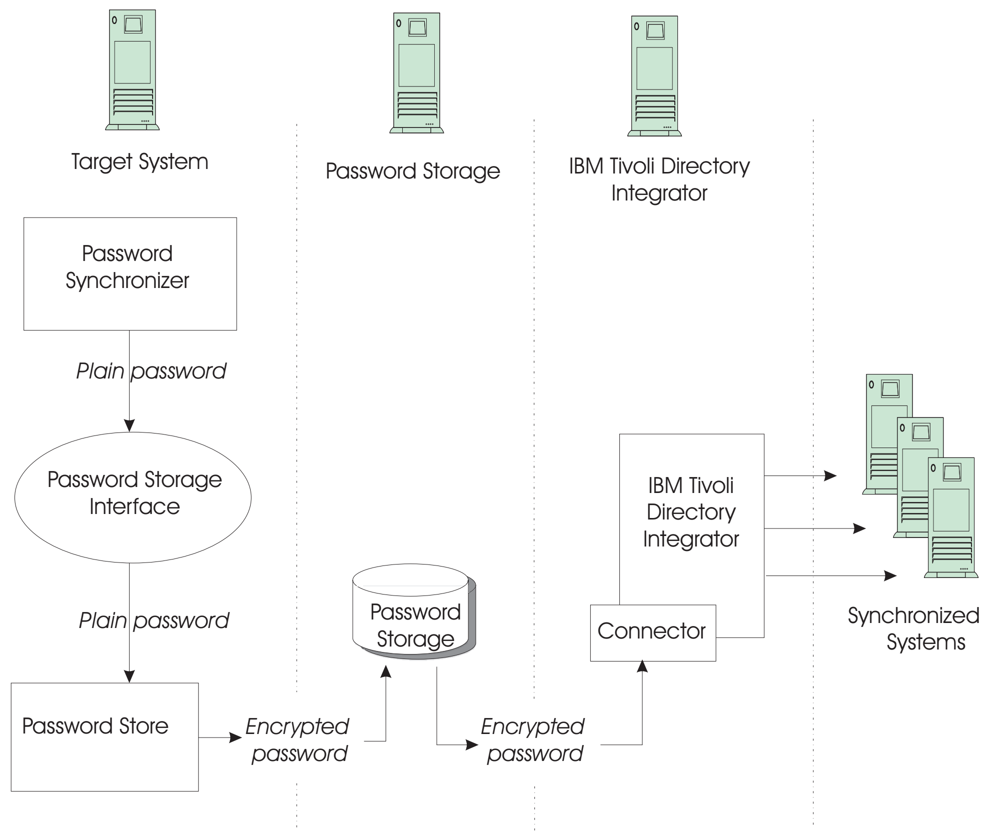


*Figure 1. Tivoli Directory Integrator Password Synchronizer architecture*

**Target System** on the diagram designates the software system where we want to intercept password changes. The **Password Synchronizer** component hooks into the Target System using custom interfaces provided by the Target System. The Password Synchronizer component intercepts password changes as they occur in the Target System and before the password is hashed irreversibly.

Also, a **Password Store** component is deployed on the Target System. Once the Password Synchronizer intercepts a password change it immediately sends the password to the Password Store. The Password Store encrypts the password and sends it to a **Password Storage**.

The Password Storage is the second layer in the architecture and represents a persistent storage system (for example, an LDAP directory, or WebSphere MQ Everyplace) where the intercepted and already-encrypted passwords are stored in a form and location that are accessible from the IBM Tivoli Directory Integrator . The Password Storage can reside on the Target System machine or on another network machine.

The third layer of the architecture is represented by the IBM Tivoli Directory Integrator . The IBM Tivoli Directory Integrator uses a Connector component to connect to the Password Storage and retrieve the passwords stored there. Once in the IBM Tivoli Directory Integrator , the passwords are decrypted and made available to the AssemblyLine that synchronizes them with other systems. The IBM Tivoli Directory Integrator can be deployed on a machine different than the Target System and Password Storage machines.

The next layer in the architecture (in the data flow direction) is represented by the systems whose passwords are synchronized with the Target System. The password synchronization AssemblyLine is responsible for connecting to these systems and updating the passwords there.

## The Password Store Interface

A key element of the IBM Tivoli Directory Integrator password synchronization architecture is the **Password Store Interface**. The Password Store Interface mediates between the Password Synchronizer and the Password Store components. Password Store components implement this interface and Password Synchronizer components use this interface to interact with the Password Stores. This enables using any Password Synchronizer with any Password Store.

Also, the Password Store used by a Password Synchronizer can be easily changed when necessary. For example, a Password Synchronizer for IBM Tivoli Directory Server is deployed and configured to use the LDAP Password Store. After time it is decided that you need to use MQe Password Store. Then you need to configure the MQe Password Store, change a single property of the Password Synchronizer, and restart the IBM Tivoli Directory Server. New password changes are stored in MQ Everyplace. It is not necessary to install the solution again.

## Architecture options

For simplicity, the previous diagram shows password interception on a single Target System. Actually, a password synchronization solution might need to intercept password changes on several Target Systems. This is where the layered password synchronization architecture brings additional value in terms of scalability and customization options:

- The Password Store components of several Target Systems can be configured to store the intercepted passwords in the same Password Storage. The IBM Tivoli Directory Integrator AssemblyLine uses a single Connector to connect to the Password Storage and is not affected by the number of Target Systems whose passwords are intercepted and stored in this Password Storage.
- The AssemblyLine can be configured to connect to several Password Storages (using several Iterator Connectors). This is useful when different Password Storages have to be used, or distinction of the Target Systems on IBM Tivoli Directory Integrator is necessary.

In either (or both) of these previous approaches, it is possible to add, remove or change Target Systems in an already existing solution by focusing mainly on the new functionality without affecting the rest of the solution.

On the other end of the data flow, where passwords are updated in systems that you want to keep synchronized, the password synchronization architecture benefits from the inherent scalability of the IBM Tivoli Directory Integrator . Updating passwords on yet another system might be as easy as adding a new Connector in the password synchronization AssemblyLine.

In the case where the Target System is also one of the systems updated with the intercepted passwords from other systems, special care must be taken to avoid circular updates. The implementation on the IBM Tivoli Directory Integrator side must build logic that does not update a system with passwords intercepted on that same system.

## Security

Public-private key infrastructure is used to provide secure transport and intermediate storage of password data.

The Password Store components use a public key to encrypt password data before sending it on the wire and storing it in the Password Storage. The IBM Tivoli Directory Integrator AssemblyLine or specialized Connectors have the corresponding private key and use it to decrypt password data retrieved from the Password Storage.

An additional layer of security is added by Password Store components supporting SSL.

The installation folder of each password synchronizer and all files in it must be protected against non-trusted users on the host operating system. The preferred way to achieve this is by setting proper file system permissions – non-trusted users and groups must not have any access (read, write, execute) to the installation folder or the files of the password synchronizer.

## Reliability

Functionality for preventing and dealing with possible password de-synchronization is built into the password synchronization workflow.

The Password Synchronizer and Password Store components together provide functionality to deal with cases where an external storage system is not available or malfunctions.

The Password Store always reports to the Password Synchronizer whether or not the password was successfully stored into the Password Storage. The Password Synchronizer component can do the following to prevent or handle possible password de-synchronizations:

- The Password Synchronizer can cancel the password change in the Target System after the Password Store reports that the password is not stored into the Password Storage (due to availability or other reasons), where enabled.
- Where the Target System does not enable cancel or rollback on the password change (which you want to do on unsuccessful storage), the failure is logged with information about the user whose password is not stored in the Password Storage. An Administrator can inspect the log and resolve de-synchronized passwords.

# Chapter 2. Installing the Password Synchronization Plug-ins

This chapter describes the installation of the Tivoli Directory Integrator Password Synchronization Plug-ins.

## Before You Install

Before you install, please read the following sections and make sure your system meets the minimum requirements.

### Disk Space Requirements

The IBM Tivoli Directory Integrator Password Synchronization Plug-ins Solution Installer requires 450 MB of temporary disk space during installation. Additionally, the following disk space amounts are required for Tivoli Directory Integrator components that will remain on the system after installation is complete.

Disk space requirements by platform for an installation in which all components are selected:

- Windows (32 and 64 bit): 170 MB
- Linux: (32 and 64 bit) 180 MB
- AIX®: 170 MB
- Solaris: 250 MB
- HP-UX: 300 MB

### Memory Requirements

The IBM Tivoli Directory Integrator Password Synchronization Plug-ins Installer requires 512 MB of memory. The precise amount of required memory after installation depends on the components you choose to install.

Most plug-ins will run within the memory space of the product relevant to the plug-in. On other words, the IBM Directory Server password catcher plug-in will execute within the memory space of the IBM Directory Server. The memory space requirements listed below for the plug-ins on each operating system platform, are in addition to the memory requirements of the integrated product.

Memory space requirements by platform for an installation in which all components are selected:

- Windows (32 and 64 bit): 128 MB
- Linux: (32 and 64 bit) 128 MB
- AIX: 128 MB
- Solaris: 128 MB
- HP-UX: 128 MB

### Platform Requirements

Platform requirements for each Password Synchronization Plug-In is documented in the “Supported Platforms” section of the chapter for each Password Synchronization Plug-In.

### Root or Administrator Privileges

On Windows platforms, the installer requires that the user ID used to install IBM Tivoli Directory Integrator Password Synchronization Plug-ins be the Administrator ID or a member of the Administrators group. On UNIX platforms, the installer requires that the user be root. The installer will fail if the user ID used to install IBM Tivoli Directory Integrator Password Synchronization Plug-ins does not have these privileges.

## Launching the appropriate installer

You can launch the IBM Tivoli Directory Integrator Password Synchronization Plug-ins Installer by using either of the following methods:

- Launch the installer from the Launchpad, or
- Launch the installer directly.

For more information about the Launchpad method, see the *IBM Tivoli Directory Integrator Administration Guide*.

To launch the installer directly, using the installation executable:

1. Locate the install executable file for your platform in the `plugins_installer` directory on the product CD:

   **Windows Intel®**
      `install_tdiv611plugins_win32.exe`

   **Windows AMD64/EM64T**
      `install_tdiv611plugins_amd64windows.exe`

   **AIX** `install_tdiv611plugins_aix.bin`

   **Linux** `install_tdiv611plugins_linux.bin`

   **Linux AMD64/EM64T**
      `install_tdiv611plugins_amd64linux.bin`

   **Solaris**
      `install_tdiv611plugins_solaris.bin`

   **HP-UX**
      `install_tdiv611plugins_HPUX.bin`

2. Double-click the executable, or type the executable name at the command prompt. This launches the installer. For information on how to use the installer, see "Using the platform-specific IBM Tivoli Directory Integrator Password Synchronization Plug-ins installer."

Once you have launched the installer (using the Launchpad or by starting the platform-dependent installer directly), you are ready to begin the process of installing Tivoli Directory Integrator.

**Note:** In order to install the product, you will require Administrator privilege.

## Using the platform-specific IBM Tivoli Directory Integrator Password Synchronization Plug-ins installer

1. The "Welcome" panel of the IBM Tivoli Directory Integrator Password Synchronization Plug-ins Installer provides you with information about IBM Tivoli Directory Integrator Password Synchronization Plug-ins. Click **Next** to continue.

2. After reading the Software license agreement, select *I accept the terms in the license agreement* if you accept its terms. Click **Next** to continue.
3. You are then prompted to specify the Destination Path for the installation. You may edit the default path shown, or use the **Browse** button to browse to the desired path. After you have specified the Destination Path, click **Next** to continue, or click **Back** to go back to the previous screen.
4. The next screen allows you to select which features to install. Each feature represents one Plug-in. You may install one or more plug-ins for installation.
5. If the plug-ins you selected to install includes the Domino Server plug-in, then the next screen will prompt you for two Domino related directories, the Domino Data Directory, and the Domino JVM (Java Virtual Machine) Directory. You need to enter a valid directory name for each of these on this screen. When you have entered both directories, click **Next** to continue, or click **Back** to go back to the previous screen.
6. The next one or more screens prompt you to select the Storage Method for each plug-in you are installing. You will be presented with one of these Storage Method selection screens for each Plug-In. On each Storage Method selection screen, you must select between using WebSphere MQ Everyplace or LDAP for password storage. After selecting the Storage Method for each Plug-In, click **Next** to continue, or click **Back** to go back to the previous screen.
7. A dialog is shown while certain system checks are performed. When that activity is complete, a screen prompts for confirmation of the installation. Click the **Install** button to install the selected Plug-Ins with the selected password stores, or click **Back** to go back and make changes to your selections.
8. Installation of your selected Plug-Ins may take several minutes to complete.

## Installing using the command line

The following command line options are supported by the IBM Tivoli Directory Integrator Password Synchronization Plug-ins Installer:

**-console**

Specifies to use the console interface mode, where messages during installation are displayed on the Java console and the wizard is run in console mode. For example:

**Windows**

```
install_wizard\install_tdiv611plugins_win32.exe –console
```

**Linux**

```
install_wizard/install_tdiv611plugins_linux.bin –console
```

**Note:** The following messages might be displayed when you install Tivoli Directory Integrator 6.1 using the **-console** option on an HP-UX operating system:

```
rm: java not removed.  Text file busy
rm: directory PA_RISC2.0 not removed.  Directory not
    empty
rm: directory bin not removed.  Directory not empty
rm: libhpi.sl not removed.  Text file busy
rm: directory native_threads not removed.  Directory
    not empty
rm: libjvm.sl not removed.  Text file busy
rm: directory server not removed.  Directory not empty
rm: libjava.sl not removed.  Text file busy
rm: libnet.sl not removed.  Text file busy
rm: libnio.sl not removed.  Text file busy
rm: libverify.sl not removed.  Text file busy
```

```
rm: libzip.sl not removed.  Text file busy
rm: directory PA_RISC2.0 not removed.  Directory not
    empty
rm: directory lib not removed.  Directory not empty
rm: directory jre not removed.  Directory not empty
rm: directory _bundledJRE_ not removed.  Directory
    not empty
rm: directory /tmp/istemp8353117173137 not removed.
    Directory not empty
```

You can safely ignore these messages, which do not cause the installer to fail.

**-options-record**
Specifies that the IBM Tivoli Directory Integrator Password Synchronization Plug-ins Installer should automatically generate a response file for the project after the completion of the installation or uninstallation. For example (one line):

**Windows**
```
install_wizard\install_tdiv611plugins_win32.exe
     –options-record response_file_name
```

**Linux**
```
install_wizard/install_tdiv611plugins_linux.bin
     –options-record response_file_name
```

**-options**
Specifies that a response file be used to execute the installation or uninstallation of IBM Tivoli Directory Integrator Password Synchronization Plug-ins. A response file is usually used when a silent. For example:

**Windows**
```
install_wizard\install_tdiv611plugins_win32.exe
   –options response_file_name
```

**Linux**
```
install_wizard/install_tdiv611plugins_linux.bin
   –options response_file_name
```

**-silent** Specifies to install or uninstall the product in silent mode, where the installation or uninstallation is performed with no user interaction. The **-options** command line option is use to specify what response file to use. For example:

**Windows**
```
install_wizard\install_tdiv611plugins_win32.exe
   –silent response_file_name
```

**Linux**
```
install_wizard/install_tdiv611plugins_linux.bin
   –silent response_file_name
```

**-is:javahome**
Specifically tells the Installer the home directory location of the Java Virtual Machine (JVM) to use. The JVM that is specified must be at the 1.4.2 level or higher. For example:

**Windows**
```
install_wizard\install_tdiv611plugins_win32.exe
   –is:javahome c:\java1.42
```

**Linux**
```
install_wizard/install_tdiv611plugins_linux.bin
   –is:javahome /opt/IBM/java1.42
```

**-is:log** Specifically tells the Installer to use the specified file for logging the installation progress. This can be useful if reporting problems to IBM support. For example:

**Windows**
```
install_wizard\install_tdiv611plugins_win32.exe
   –is:log c:\temp\log.txt
```

**Linux**
```
install_wizard/install_tdiv611plugins_linux.bin
   –is:log /tmp/log.txt
```

**-is:silent**
Prevents the display of the Launcher UI to the end user. This does not launch the application in silent mode (prevent the display of the wizard). Use the **-silent** option to run in silent mode.

**-is:tempdir**
Specifically tells the Installer to use the specified directory to which for writing temporary files. If the specified directory does not exist, the Installer will use the system temp directory. For example:

**Windows**
```
install_wizard\install_tdiv611plugins_win32.exe
   –is:tempdir c:\privateTemp
```

**Linux**
```
install_wizard/install_tdiv611plugins_linux.bin
   –is:tempdir /privateTemp
```

## Performing a silent install

To perform a silent installation you must first generate a response file. To generate this file, perform a non-silent install with the -options-record option specified. For example:

**Windows**
```
install_wizard\install_tdiv611plugins_win32.exe
   –options-record response_file_name
```

**Linux**
```
install_wizard/install_tdiv611plugins_linux.bin
   –options-record response_file_name
```

The response file is created in the directory that you specify during installation. Once the response file is created, you can install silently using the following command:

**Windows**
```
install_wizard\install_tdiv611plugins_win32.exe
   –silent –options response_file_name
```

**Linux**
```
install_wizard/install_tdiv611plugins_linux.bin
   –silent –options response_file_name
```

## Uninstalling

You can uninstall IBM Tivoli Directory Integrator Password Synchronization Plug-ins entirely or uninstall certain components.

## Launching the uninstaller

To uninstall IBM Tivoli Directory Integrator Password Synchronization Plug-ins, you must first launch the uninstaller:

1. Navigate to the IBM Tivoli Directory Integrator Password Synchronization Plug-ins `_uninst_plugin` directory, for example:

   ```
   install_path/_uninst_plugin
   ```

2. Launch the uninstaller by executing the uninstall executable:

   **Windows**
   ```
   uninstall.exe
   ```

   **All other platforms**
   ```
   uninstall.bin
   ```

## Uninstalling all IBM Tivoli Directory Integrator Password Synchronization Plug-ins components

To uninstall all IBM Tivoli Directory Integrator Password Synchronization Plug-ins components:

1. On the "Welcome" panel, select the **Remove** radio button. Click **Next**.
2. Verify that you want to uninstall Integrator Password Synchronization Plug-ins. Click **Back** to make any changes. When you are ready to uninstall, click **Uninstall**.
3. When the uninstall completes, click **Finish**.

## Uninstalling individual IBM Tivoli Directory Integrator Password Synchronization Plug-ins

To uninstall certain IBM Tivoli Directory Integrator Password Synchronization Plug-ins components:

1. On the "Welcome" panel, select the **Remove Features** radio button. Click **Next**.
2. Installed features are indicated by check marks. To uninstall a feature, click the check mark to deselect it. Click **Next** to continue.
3. Verify the list of features to be removed under Features to Remove. Click **Back** to make any changes. When you are ready to uninstall, click **Uninstall**.
4. When the uninstall completes, click **Finish**.

## Performing a silent uninstall

To perform a silent uninstall, you must first generate a response file. To generate this file, perform a full GUI uninstall with the `-options-record` option specified. For example (entered as one line):

**Windows**
```
install_path\_uninst_plugin/uninstall.exe –options-record
     uninstall_response_file_name
```

**Linux**
```
install_path/_uninst_plugin/uninstall.bin –options-record
     uninstall_response_file_name
```

The response file is created in the directory that you specify during uninstallation. Once the response file is created, you can uninstall silently using the following command:

**Windows**

```
install_path\_uninst_plugin/uninstall.exe -silent -options
     uninstall_response_file_name
```

**Linux**

```
install_path/_uninst_plugin/uninstall.bin -silent -options
     uninstall_response_file_name
```

## Upgrading the Password Synchronization Plug-ins

There is no upgrade functionality of the IBM Tivoli Directory Integrator Password Synchronization Plug-ins Installer. Upgrading from a previous version of the IBM Tivoli Directory Integrator Password Synchronization Plug-ins is achieved by uninstalling the previous version, and installing the new version.

**Note:** Upgrading is not relevant to the Password Synchronizer for UNIX and Linux.

To upgrade Password Synchronization Plug-ins:

1. Make a backup of any file you modified for the previous version.
2. Navigate to the IBM Tivoli Directory Integrator Password Synchronization Plug-ins `_uninst` directory for the previous version. For example:

   `install_path/_uninst`
3. Launch the uninstaller by executing the uninstall executable.

   **Windows**

   `uninstall.exe`

   **All other platforms**

   `uninstall.bin`
4. Reboot the system.
5. Install the new version of Password Synchronization Plug-ins following the instructions given above.
6. Restore the files backed up in step 1.

# Chapter 3. Windows Password Synchronizer

This chapter describes the Tivoli Directory Integrator Windows Password Synchronization Plug-in.

This chapter includes the following sections:

## Overview

The Password Synchronizer for Windows intercepts password changes of user accounts on Windows operating systems.

Password changes are intercepted in all of the following cases:

- When a user changes his own password through the Windows user interface
- When an administrator changes the password of a user through the Windows administrative user interface
- When a password change request to Active Directory is made through LDAP

The IBM Directory Integrator Password Synchronizer plug-in propagates the changes to a repository (Password Store) before the Windows system changes the password.

The IBM Directory Integrator Password Synchronizer stores the user password in a Password Store (LDAP server, WebSphere MQ Everyplace).

The change is later propagated to other servers by an IBM Directory Integrator AssemblyLine. After the password is successfully stored in the Password Store, control is returned to the Windows system and the user password is modified.

### Synchronizing from a single machine

To synchronize passwords from a single machine, install the Password Synchronizer on the Windows standalone machine.

### Synchronizing from a Windows 2000, Windows XP or Windows 2003 domain

To synchronize password changes from a Windows 2000, Windows XP or Windows 2003 domain, install the Password Synchronizer on all domain controllers for the domain with which you want to synchronize.

### Sample scenario

Bob logs onto the windows machine, presses **Ctrl**+**Alt**+**Delete**, and requests a password change. That password change is intercepted by the Password Synchronizer, then delegated to the associated Password Store (LDAP Password Store, MQe Password Store). If the Password Store confirms that the password was successfully stored, then the password change takes place on the native Windows machine, whether it is a standalone machine or a domain controller. If the

Password Store indicates that the password was not stored, then the password change on the native Windows machine is denied.

Password change requests to Active Directory through LDAP/JNDI are also intercepted and handled by the Password Synchronizer.

## Windows Password Synchronizer Workflow

The Windows Password Synchronizer intercepts a password change before the change is actually committed internally by Windows and Active Directory. The Password Synchronizer passes the new password to the Password Store.

If the Password Store indicates that the password is stored successfully, the Password Synchronizer enables the password change to be committed in Windows.

If the Password Store indicates that the password is not stored, the password change is rejected on the Windows machine. If the password change has been performed from the Windows user interface, an error box is displayed with contents similar to:

```
Windows cannot complete the password change for <user_name> because:
The password does not meet the password policy requirements.
Check the minimum password length, password complexity and password history
requirements.
```

This is a standard message that is displayed by Windows when the password change is denied. The log files of the Password Synchronizer and the Password Store component indicate the actual reason why the password cannot be stored in the Password Storage.

## Windows Password Synchronizer Filtering

This version of the Windows Password Synchronizer has introduced filtering functionality. Filtering affects only whether a password change is sent to the password store and not whether the Windows domain accepts or rejects the password change. If the user filter accepts a user, the password changes for that user are sent to the password store. Otherwise, password changes for that user are not sent to the password store for that user.

The user filter makes decisions based on two factors/criteria.

- group membership,
- DN matching (which is in effect LDAP sub-tree location matching).

Group membership deals with whether a user is a member of some Windows group. The user filter does not recognize nested groups, so if a user is a member of group A, which is nested into group B, then the user will not be deemed member of group B.

DN matching deals with whether a DN suffix matches the Distinguished Name of a user. For example if the user has a Distinguished Name `cn=myuser,ou=myou,dc=mydc,dc=com` it is matched by the DN suffix `dc=mydc,dc=com` but not by `dc=mydc`.

The user filter allows include and exclude rules of both group membership and DN matching. For example the user filter can be configured to accept all users which are members of a certain Windows group (include form) but not members of some other Windows group (exclude form).

Group membership and DN matching in both rules (include/exclude) can be freely combined. However, there is one specific limitation. Exclude rules always have higher priority than include rules. So for example if a user is included by DN matching but excluded by group membership, the user will not be accepted by the user filter.

To preserve backward compatibility, if no include form is specified (neither group membership, nor DN matching), the default form is *include all*. Inversely this also implies that if no exclude form is specified (neither group membership, nor DN matching), the default form is *exclude none*.

Here are some examples to help clarify the filtering mechanism:
- If no configuration is provided to the user filter, it accepts all users (backward compatibility).
- If the user filter is provided with some include rules and no exclude rules, it accepts only users matched by the provided include rules.
- If the user filter is provided with some exclude rules and no include rules, it accepts only users, which are NOT matched by any of the exclude rules.
- If the user filter is provided with some include rules and some exclude rules, it accepts only users, which are matched by some of the include rules and are not matched by any of the exclude rules.

The user filter of the Windows Password Interceptor is configured using 4 string values in the Windows registry to define the include and exclude rules. These values are optional and are located under the registry key of the Windows Password Interceptor:

**includeGroups**
A list of Windows groups. If a user is a member of some group on the list, the user will be accepted by the user filter (assuming the user is not excluded by some of the exclude lists).

**excludeGroups**
A list of Windows groups. If a user is a member of some group on the list, the user will not be accepted by the user filter.

**includeDNs**
A list of DN suffixes. If a user's Distinguished Name matches some suffix on the list, the user will be accepted by the user filter (assuming the user is not excluded by some of the exclude lists).

**excludeDNs**
A list of DN suffixes. If a user's Distinguished Name matches some suffix on the list, the user will not be accepted by the user filter.

All of the above registry string values must be lists, whose tokens are separated with semicolons. Redundant white-spaces are not allowed. The group lists must include only names of existing Windows groups. Matching of a DN suffix against a Distinguished Name is performed by a simple case-insensitive string comparison – no special treatment is provided for white-spaces. For example, the `DC=COM` suffix matches the `cn=myuser,dc=mydc,dc=com` Distinguished Name, but the `dc = com` suffix does not.

If the user filter mechanism encounters an issue (for example an invalid group name in its configuration), an error message is logged and the Windows Password Interceptor acts as if the filter has accepted the user. If the user filter decides to not accept a user, a message stating that is logged.

The configuration of the user filter is read again on each password notification, so changes to the configuration have immediate effect – there is no need to restart the Windows operating system for the changes to the user filter to be taken into account.

**Note:** The user filter configuration of the Windows Password Synchronizer is sensitive to modifications of the Windows groups, involved in the configuration. If some of the following changes occur, the Windows Password Synchronizer must be restarted (which requires restart of the operating system):

- the Windows name of a group is modified (this corresponds to the `sAMAccountName` attribute in Active Directory),
- the distinguished name of a group is modified (for example the group is moved to another container).

This restriction holds true for all groups, which have appeared in the configuration of the Windows Password Synchronizer during its lifetime.

**Attention:** The user filtering feature of the Windows Password Synchronizer will function properly only on machines, which are part of a Windows domain. Workgroup machines will not be able to use user filtering. If one configures user filtering on a workgroup machine, the plug-in will log an error message like the following on every password change and will send the change to the password store, no matter the provided configuration:

```
User filtering failed: The specified domain either does not exist or could not
   be contacted.
```

Note that if no user filtering configuration is supplied, the plug-in will function normally and no error will be logged (because no filtering is performed).

## Supported Platforms

The following platforms are supported for the Tivoli Directory Integrator Windows Password Synchronization Plug-in:

- Windows 2000 Professional (IA32)
- Windows 2000 Server (IA32)
- Windows 2000 Advanced Server (IA32)
- Windows XP Professional (IA32)
- Windows 2003 Standard Edition (IA32)
- Windows 2003 Enterprise Edition (IA32)
- Windows 2003 Standard Edition (AMD64/EMT64)
- Windows 2003 Enterprise Edition (AMD64/EMT64)

## Deployment and Configuration

### Prerequisites

The IBM Directory Integrator Password Synchronizer requires JRE 1.5 (included).

### Included files

**tdipwflt611.dll**
Contains the implementation which interfaces with Windows password change function for 32-bit Windows environments.

**tdipwflt611_64.dll**
: Contains the implementation which interfaces with Windows password change function for 64-bit Windows environments.

  **Note:** In a 64 bit Windows environment, `tdipwflt61_64.dll` needs to be renamed to `tdipwflt61.dll` and placed in the Windows system folder.

**idiMQE.reg**
: Used to create registry entries required for the MQe Password Store.

**idiLDAP.reg**
: Used to create registry entries required for the LDAP Password Store.

**setWinPwSyncAccTypes.reg**
: Used to define the type of Windows accounts whose password changes will be intercepted.

**pwsync_admin.exe**
: Administration Tool executable for supported 32bit Windows platforms.

**pwsync_admin_64.exe**
: Administration Tool executable for supported 64bit Windows platforms.

**idicryptokeys.bat**
: Utility batch file that creates test/demo keystore file.

**mqeconfig.props**
: Configuration file required when using the Mqe Password Store.

**JAR Files**
: Contain the Password Synchronizer functionality that interfaces with the LDAP and MQe Password Stores: `proxy.jar`, `stopproxy.jar`, `mqepwstore.jar`, `idipwstore.jar`, `idipwcrypto.jar`, `mqeconfig.jar`, `auibase.jar`, `tdipwflt.jar`, `ibmjms.jar`, `MQeBase.jar`, `MQeJMS.jar`, `MQeSecurity.jar`, `MQeJMX.jar`.

**LDIF Files**
: Contain the LDAP schema definition required for the specific Directory Server used as the LDAP Password Store: `ibm-diPersonSchemaForAD.ldif`, `ibm-diPersonForSunDS.ldif` , `ibm-diPerson_oc.ldif`, `ibm-diPerson_z.ldif`.

## Verifying registry settings

The IBM Tivoli Directory Integrator Password Synchronization Plug-ins Installer creates the following registry entries in the key folder:

```
HKEY_LOCAL_MACHINE\SOFTWARE\IBM\Tivoli Directory Integrator 6.1.1\Windows
    Password Synchronizer
```

The following keys should exist with the corresponding value:

```
Class: REG_SZ: com.ibm.di.plugin.idipwsync.IDIPasswordSynchronizer

Classpath: REG SZ: "c:\<install_directory>"

Java:REG_SZ: "c:\<install_directory>\jvm\jre\bin\java.exe"
```

To verify the settings using **regedit**, select **Edit** then **Find**. Enter **Windows Password Synchronizer**. Click **find next**. You see something like the following (for the “Windows Password Synchronizer” key) in the key folder :

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\LSA\Windows Password
   Synchronizer
```

The following key should exist with the corresponding value:

```
Notification Packages: REG_MULTI_SZ: tdipwflt61
```

## Setting the Password Synchronization Account Types

The IBM Tivoli Directory Integrator Password Synchronization Plug-ins Installer creates most of the required Windows registry entries. There is, however, one registry key which you must set manually - it is called ″AccountTypes″. The Password Synchronizer plug-in is capable of reporting password changes to the following Windows account types:

**NORMAL_ACCOUNT**
This is a default account type that represents a typical user.

**TEMP_DUPLICATE_ACCOUNT**
This is an account for users whose primary account is in another domain.

**INTERDOMAIN_TRUST_ACCOUNT**
This is a permit to trust account for a domain that trusts other domains.

**WORKSTATION_TRUST_ACCOUNT**
This is a computer account for a computer that is a member of this domain.

**SERVER_TRUST_ACCOUNT**
This is a computer account for a backup domain controller that is a member of this domain.

The ″AccountTypes″ key value is stored in the registry key folder:

```
HKEY_LOCAL_MACHINE\SOFTWARE\IBM\Tivoli Directory Integrator 6.1.1\Windows
   Password Synchronizer\AccountTypes
```

The value of this parameter is a character string, indicating the account types for which the Password Synchronizer will report password changes. Its format is a space-delimited list of account types.

An example value for this key would be:

```
"NORMAL_ACCOUNT WORKSTATION_TRUST_ACCOUNT"
```

**Note:** The Password Synchronizer always reports password changes to accounts of type NORMAL_ACCOUNT regardless of whether NORMAL_ACCOUNT is specified in the AccountTypes parameter.

To set the ″AccountTypes″ key in the Windows registry:

1. Locate the Windows registry file `setWinPwSyncAccTypes.reg`. This default file is copied to the plug-in installation folder by the distribution package installation program.
2. The default setting for the ″AccountTypes″ key in the `setWinPwSyncAccTypes.reg` file is ″NORMAL_ACCOUNT″. This means that the Password Synchronizer will only report password updates to normal user accounts. If you wish to change this setting:
   a. Edit the `setWinPwSyncAccTypes.reg` file in a text editor.
   b. Change the default setting of the ″AccountTypes″ key from ″NORMAL_ACCOUNT″ to the desired setting.
   c. Save and close the `setWinPwSyncAccTypes.reg` file.
3. Double-click the `setWinPwSyncAccTypes.reg` file in Windows Explorer. This sets the ″AccountTypes″ key at the correct location in the Windows registry.

4. Reboot the Windows machine so that the changes can take effect.

## Enabling Local Security

Change the Local Security Policy as follows:

1. Select **Control Panel**-->**Administrative Tools**-->**Local Security Policy**
2. Select **Account Policies**-->**Password Policy**
3. Change **Passwords must meet complexity requirements** to **enabled**.

**Note:** For this change to take place, reboot the machine. Make sure that you set up the Password Store properties file before rebooting the machine.

## Password Stores setup information

During installation you will be prompted to choose a Password Store. The installer will configure the Password Synchronizer to use the chosen Password Store, but will not configure the Password Store itself.

For information on setting up the Password Stores, see the following resources:

- Chapter 8, "LDAP Password Store," on page 73.
- Chapter 9, "MQ Everyplace Password Store," on page 85.

## Changing the Password Store

The Password Store used by the Windows Password Synchronizer can be changed at any time after the initial deployment of the solution.

To switch the Windows Password Synchronizer to use the LDAP Password Store:

1. Make sure the LDAP Password Store is configured.
2. Double-click on the file `idiLDAP.reg` (placed in the install directory of the Windows Password Synchronizer).
3. Click **Yes** to change the registry settings.
4. Restart the machine.

To switch the Windows Password Synchronizer to use the MQe Password Store:

1. Make sure the MQe Password Store is configured.
2. Double-click on the file `idiMQE.reg` placed in the install directory of the Windows Password Synchronizer.
3. Click **Yes** to change the registry settings.
4. Restart the machine.

## Trace Logging

The Windows Password Synchronizer supports four types of trace logging:

- Information,
- Diagnostic,
- Warning,
- Error.

To enable each of the levels of trace logging, set the respective logging level property to *true* within the `idipwsync.props` file. An extract from `idipwsync.props` showing the all trace logging enabled is show below:

```
InformationTrace=true
DiagnosticTrace=true
WarningTrace=true
ErrorTrace=true
```

Once trace logging is enabled, the Windows Password Synchronizer will produce the following trace files when the synchronized machine is restarted:

- `C:\PWFLTTC.trc`, which contains trace messages generated while the Windows Password Synchronizer is being initialized.
- `C:\PWFLTOUT.trc`, which contains Information and Diagnostic trace information.
- `C:\PWFLTERR.trc`, which contains Warning and Error trace information.

## Plug-in administration tool

A command line tool for performing administrative tasks, `pwsync_admin.exe`, can be found in the plug-in installation directory. The primary purpose of this administrative tool is to allow reconfiguration of the Windows Password Synchronizer without rebooting the Windows machine. For example, this tool enables changing of the password store without rebooting Windows.

**Note:** The only change that cannot be accomplished without rebooting Windows is replacing the `tdipwflt.dll` plug-in, located in the Windows `System32` directory.

### Usage

This is how the administration tool is used from the command line:

```
pwsync_admin.exe command for 32 bit Windows
pwsync_admin_64.exe command for 64 bit Windows
```

This tool takes a single command line parameter (the command argument above), whose value can be one of the following:

**suspend_plugin**
This command writes a boolean value to the Windows registry (please see the Windows registry settings section), thus indicating to the plug-in that subsequent password changes must not be propagated to the Java proxy. This command causes subsequent password changes to be skipped until a **resume_plugin** command is issued.

**resume_plugin**
This command writes a boolean value to the Windows registry (please see the Windows registry settings section), thus indicating to the plug-in that subsequent password changes must be propagated to the Java proxy. This command causes subsequent password changes to be synchronized until a **suspend_plugin** command is issued.

**stop_proxy**
This command causes the administration tool to connect through a socket to the command socket port of the Java proxy and send a stop request to the proxy. This causes the proxy to terminate gracefully.

**start_proxy**
This command starts the Java proxy, which causes the proxy configuration to be reloaded.

**restart_proxy**
This command is equivalent to a **stop_proxy** command followed by a **start_proxy** command.

## Windows registry settings

The Java proxy listens on a server socket for administrative commands to be sent by the administration tool. The port of the command socket can be specified in the Windows registry in the following key:

```
 [HKEY_LOCAL_MACHINE\SOFTWARE\IBM\Tivoli Directory Integrator 6.1.1\Windows
     Password Synchronizer] "ProxyCommandPort"="19010"
```

If this key is missing, a default value of 19001 will be used. If Tivoli Directory Integrator server is installed and running on the same system, then port 19001 can not be used for the Windows Password Synchronizer.

The value of this key is used by both the Java proxy and the administration tool - it lets the administration tool connect to the Java proxy by using the same port. This key can only be created manually, but creating this key is only required when the Java proxy command port needs to be changed.

**Note:** Changing this key should be done only when the Java proxy is stopped. Otherwise on the next invocation of the administration tool it will not be able to connect to the Java proxy.

The registry key used by the **suspend_plugin** and **resume_plugin** commands is:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\IBM\Tivoli Directory Integrator 6.1.1\Windows
     Password Synchronizer]
"disabled"="true"
```

If the key has a value of *true*, then the plug-in will not synchronize passwords. If this key is missing or has a value other than *true*, the plug-in will synchronize passwords. This key is created by the plug-in administration tool on first use.

**Note:** Neither of these two keys is present in the Windows registry after the plug-in is installed. These keys are not required for the normal operation of the plug-in.

## Logging

The administrative tool logs messages both to the console and to a log file named `pwsync_admin.log`, which is located in the install directory of the plug-in. The log file can be used for analyzing errors encountered during administrative tool operations, or an historical reference for operations performed using this tool.

## Considerations when using the administration tool

When using the administration tool, be aware of the following considerations:

- When the plug-in is suspended, password changes are skipped (not propagated) by the plug-in. This can result in inconsistencies (password changes lost) in the target synchronization system
- When the Java proxy is started, it loads the password store configuration file. This happens when the machine is rebooted, or when the plug-in is not suspended but the Java proxy is stopped as a password change occurs. If the user is editing the configuration file at the time, the Java proxy may load a possibly corrupted configuration.
- When the plug-in is not suspended and the Java proxy is not running, if a password change is issued with the **Active Directory Users and Computers** user interface tool, the plug-in is notified by Windows two or three times of this password change. The result is that the same password update is propagated two or three times. This happens because the plug-in starts the proxy on the next password change, which takes some time. This causes Windows to notify

the plug-in several times of the same password change. This multiple reporting, however, is only present the first time the Java proxy is not running, because on subsequent password changes the Java proxy is already running.
- When the plug-in is configured with the LDAP Password Store and the LDAP Store itself is set for asynchronous storing (`waitForStore=false` specified in the LDAP Store configuration file), and when the plug-in is not suspended, it is possible that a **stop_proxy** command would cause some password changes to be skipped.

The following recommendations help address these problems:
- Suspend the plug-in using a **suspend_plugin** command prior to any **stop_proxy** or **restart_proxy** commands.
- Make a copy of the configuration file for editing purposes. Replace the old configuration file with the new one when all edits are complete.
- Make any necessary configuration changes at a low usage time, so that few (if any) password changes will be skipped and not propagated.

## Example for changing the configuration without rebooting the Windows machine

The following steps show how the log file of the MQe password store can be changed without rebooting the Windows machine:

**Note:** After these steps are completed the MQe password store will start logging to the new log file. During the short window when the plug-in is suspended, however, password changes could be skipped. They will occur in the Windows domain controller, but they will not be propagated by the plug-in. Therefore, this procedure should occur at a low usage time, when password changes are unlikely.

1. Copy the **mqepwstore.props** configuration file of the MQe password store to a temporary location.
2. Edit the file in this temporary location. Change the value of the **logFile** property.
3. Copy the edited file back to the original location.
4. Run the **pwsync_admin.exe suspend_plugin** command.
5. Run the **pwsync_admin.exe stop_proxy** command.
6. Run the **pwsync_admin.exe start_proxy** command.
7. Run the **pwsync_admin.exe resume_plugin** command.

# Chapter 4. Sun ONE Directory Server Password Synchronizer

This chapter describes the Tivoli Directory Integrator Sun ONE Directory Server Password Synchronization Plug-in.

This chapter includes the following sections:

## Overview

The Sun ONE Directory Server Password Synchronizer intercepts changes to LDAP passwords in Sun ONE Directory Server.

Passwords in Sun ONE Directory Server are stored in the `userPassword` LDAP attribute. The Password Synchronizer intercepts updates of the `userPassword` LDAP attribute.

The Sun ONE Directory Server Password Synchronizer intercepts modifications of the `userPassword` attribute of entries of any object class.

Password updates are intercepted for the following types of entry modifications:

- When a new entry is added in the directory and the entry contains the `userPassword` attribute.
- When an existing entry is modified and one of the modified attributes is the `userPassword` attribute. This includes the following cases:
  - The `userPassword` attribute is added (for example, the entry did not have a `userPassword` attribute before)
  - The `userPassword` attribute is modified (for example, the entry had this attribute and its value is now changed)
  - The `userPassword` attribute is deleted from the entry

**Notes:**

1. Deletion of entries is not intercepted by the Sun ONE Directory Server Password Synchronizer even when the entry contains the `userPassword` attribute.
2. The `userPassword` attribute in Sun ONE Directory Server is multiple-valued. Users might have several passwords. The Sun ONE Directory Server Password Synchronizer intercepts and reports any change of any of the password values.

## Supported platforms

The Sun ONE Directory Server Password Synchronizer is available for the Sun ONE Directory Server on the following platforms:

- Windows 2000 Server (IA32), Sun ONE DS 5.1 SP4 (32-bit) and Sun ONE 5.2 (32-bit)

- Windows 2000 Advanced Server (IA32), Sun ONE DS 5.1 SP4 (32-bit) and Sun ONE 5.2 (32-bit)
- Windows 2003 Standard Edition (IA32), Sun ONE 5.2 (32-bit)
- Windows 2003 Enterprise Edition (IA32), Sun ONE 5.2 (32-bit)
- AIX 5.2 (5200-80) (32-bit), Sun ONE 5.2 (32-bit)
- Solaris 9 (32-bit), Sun ONE DS 5.1 SP4 (32-bit) and Sun ONE 5.2 (32-bit)
- Solaris 9 (64-bit), Sun ONE 5.2 (64-bit)
- Solaris 10 (64-bit), Sun ONE DS 5.1 SP4 (32-bit) and Sun ONE 5.2 (32-bit and 64-bit)
- HP-UX 11iv2 (11.23) (32-bit), Sun ONE DS 5.1 SP4 (32-bit) and Sun ONE 5.2 (32-bit)
- HP-UX 11iv2 (11.23) (64-bit), Sun ONE DS 5.1 SP4 (32-bit), Sun ONE 5.2 (32-bit) and Sun ONE 5.2 (64-bit)

## Deployment

This section describes the steps required to install the plug-in on a Windows, UNIX or Linux platform.

### Deploying the Plug-in on Windows

#### Included Files

The following files are included with the plug-in distribution:

**sunpwsync.dll**
Contains the implementation which interfaces with Solaris account password change function for 32-bit Windows environments.

**sunpwsync.props**
Configuration file required when using the LDAP Password Store.

**sunpwsync_desc.props**
Configuration file that can be used when registering the Sun ONE Directory Server.

**idicryptokeys.bat**
Utility batch file that creates test/demo keystore file.

**mqeconfig.props**
Configuration file required when using the Mqe Password Store.

**JAR Files**
Contains the Password Synchronizer functionality that interfaces with the LDAP and MQe Password Stores: `proxy.jar`, `stopproxy.jar`, `mqepwstore.jar`, `idipwstore.jar`, `idipwcrypto.jar`, `mqeconfig.jar`, `auibase.jar`, `tdipwflt.jar`, `ibmjms.jar`, `MQeBase.jar`, `MQeJMS.jar`, `MQeSecurity.jar`, `MQeJMX.jar`.

**LDIF Files**
Contain the LDAP schema definition required for the specific Directory Server used as the LDAP Password Store: `ibm-diPersonSchemaForAD.ldif`, `ibm-diPersonForSunDS.ldif` , `ibm-diPerson_oc.ldif`, `ibm-diPerson_z.ldif`.

#### Registering the Sun ONE Directory Server Password Synchronization Plug-ins within Sun ONE Directory Server

To register the plug-in, stop Sun ONE Directory Server and add the following to the Sun ONE Directory Server configuration file:

*install_directory*\Servers\slapd-*hostname*\config\dse.ldif :

```
dn: cn=IBM DI PassSync Pre,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: IBM DI PassSync Pre
nsslapd-pluginPath: C:/install_directory/sunpwsync.dll
nsslapd-pluginInitfunc: PWSyncInitPreOp
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: C:/install_directory/
nsslapd-pluginId: ibmdi-pwsync-pre
nsslapd-pluginVersion: 6.1.1.1
nsslapd-pluginVendor: IBM
nsslapd-pluginDescription: IBM Tivoli Directory Integrator post-operation
   plug-in for password synchronization


dn: cn=IBM DI PassSync Post,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: IBM DI PassSync Post
nsslapd-pluginPath: C:/install_directory/sunpwsync.dll
nsslapd-pluginInitfunc: PWSyncInitPostOp
nsslapd-pluginType: postoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: C:/install_directory/
nsslapd-pluginId: ibmdi-pwsync-post
nsslapd-pluginVersion: 6.1.1
nsslapd-pluginVendor: IBM
nsslapd-pluginDescription: IBM Tivoli Directory Integrator post-operation
   plug-in for password synchronization
```

**Note:** *install_directory* is a variable. Do not copy and paste this example without first replacing variables with accurate data.

## Configuring the Sun ONE Directory Server Password Synchronization Plug-in

The configuration file C:\*install_directory*\sunpwsync.props contains properties in the format <*property_name*>=<*property_value*>, with each property placed on a single line.

Specify the following properties (property names are case sensitive):

**jvmPath**
The path to the Java Virtual Machine executable (`java.exe`), used to run the Java layer (for example, `C:\`*`install_directory`*`\jvm\jre\bin`).

**jvmClassPath**
The Java classpath of the Java layer. Contains the path to the JAR files needed by the Password Synchronization Plug-in and the Password Store.

**syncClassName**
The custom synchronization class that implements the **com.ibm.di.plugin.pwsync.IPasswordSynchronizer** interface and stores the passwords.

**serverPort**
The port on which the C and Java layers communicate.

**logFile**
The C layer log file, used to log synchronization failures.

**syncBase**
: The branch (base) of the directory tree where the Sun ONE Directory Server Password Synchronization Plug-in listens for password updates.

**javaLogFile**
: The log file of the Java layer.

**checkRepository**
: Specify **true** or **false** to correspondingly turn on or off the check for repository availability. When this option is turned on and the repository is not available, the Password Synchronization Plug-in cancels the password update in Sun ONE Directory Server.

The following is a (Windows) sample of the Sun ONE Directory Server Password Synchronization Plug-in configuration file, `sunpwsync.props`:

```
jvmPath=install_dir\jvm\jre\bin
jvmClassPath=install_dir\SunOne;install_dir\SunOne
   \jars\proxy.jar;install_dir\SunOne\jars\ibmjms.jar;install_dir
   \SunOne\jars\idipwcrypto.jar;install_dir\SunOne\jars\idipwstore.jar;
   install_dir\SunOne\jars\auibase.jar;install_dir\SunOne\
   jars\MQeBase.jar;install_dir\SunOne\jars\mqeconfig.jar;install_dir
   \SunOne\jars\MQeJMS.jar;install_dir\SunOne\jars\mqepwstore.jar;
   install_dir\SunOne\jars\MQeSecurity.jar;install_dir\
   SunOne\jars\stopproxy.jar;install_dir\SunOne\jars\tdipwflt.jar;
   install_dir\IDS\jars\MQeJMX.jar
syncClassName=com.ibm.di.plugin.pwsync.LDAPPasswordSynchronizer
serverPort=19013
logFile=install_dir\SunOne\sun_pw_sync.log
syncBase=dc=ibm,dc=com
javaLogFile=install_dir\SunOne\sun_pws_java.log
checkRepository=true
```

**Note:** *install_dir* is a variable. Do not copy and paste this example without first replacing variables with accurate data.

## Deploying the Plug-in on UNIX and Linux

### Included Files

The following files are included with the distribution package:

**libsunpwsync.so**
: Contains the implementation which interfaces with Solaris account password change function for 32-bit Linux and Solaris UNIX environments.

**libsunpwsync.a.so**
: Contains the implementation which interfaces with Solaris account password change function for 32-bit AIX UNIX environments.

**libsunpwsync.sl**
: Contains the implementation which interfaces with Solaris account password change function for 32-bit HP-UX UNIX environments.

**libsunpwsync_64.so**
: Contains the implementation which interfaces with Solaris account password change function for 64-bit Solaris UNIX environments.

**sunpwsync.props**
: Configuration file required when using the LDAP Password Store.

**sunpwsync_desc.props**
: Configuration file that can be used when registering the Sun ONE Directory Server.

**idicryptokeys.sh**
: Utility script file that creates test/demo keystore file.

**mqeconfig.props**
: Configuration file required when using the MQe Password Store.

**JAR Files**
: Contains the Password Synchronizer functionality that interfaces with the LDAP and MQe Password Stores: `proxy.jar`, `stopproxy.jar`, `mqepwstore.jar`, `idipwstore.jar`, `idipwcrypto.jar`, `mqeconfig.jar`, `auibase.jar`, `tdipwflt.jar`, `ibmjms.jar`, `MQeBase.jar`, `MQeJMS.jar`, `MQeSecurity.jar`, `MQeJMX.jar`

**LDIF Files**
: Contain the LDAP schema definition required for the specific Directory Server used as the LDAP Password Store: `ibm-diPersonSchemaForAD.ldif`, `ibm-diPersonForSunDS.ldif` , `ibm-diPerson_oc.ldif`, `ibm-diPerson_z.ldif`.

## Registering the Sun ONE Directory Server Password Synchronization Plug-ins within Sun ONE Directory Server

To register the plug-in, stop Sun ONE Directory Server and add the LDAP POST and PRE operation stanzas listed below to the Sun ONE Directory Server configuration file *install_dir*/Servers/slapd-*hostname*/config/dse.ldif. Locate the new stanzas directly above the first **nsSlapdPlugin** preoperation stanza, as found in the file by searching from top to bottom.

```
dn: cn=IBM DI PassSync Pre,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: IBM DI PassSync Pre
nsslapd-pluginPath: /install_directory/SunOne/libsunpwsync.so
nsslapd-pluginInitfunc: PWSyncInitPreOp
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: /install_directory/SunOne
nsslapd-pluginId: ibmdi-pwsync-pre
nsslapd-pluginVersion: 6.1.1
nsslapd-pluginVendor: IBM
nsslapd-pluginDescription: IBM Directory Integrator pre-operation
   plug-in for password synchronization


dn: cn=IBM DI PassSync Post,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: IBM DI PassSync Post
nsslapd-pluginPath: /install_directory/SunOne/sunpwsync.so
nsslapd-pluginInitfunc: PWSyncInitPostOp
nsslapd-pluginType: postoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: /install_directory/SunOne
nsslapd-pluginId: ibmdi-pwsync-post
nsslapd-pluginVersion: 6.1.1
nsslapd-pluginVendor: IBM
nsslapd-pluginDescription: IBM Directory Integrator post-operation
   plug-in for password synchronization
```

**Note:** *install_directory* is a variable. Do not copy and paste this example without first replacing variables with accurate data.

For HP-UX systems, change the extension from .so to .sl in the **nsslapd-pluginPath** entries. For example:

```
nsslapd-pluginPath: /install_dir/SunOne/sunpwsync.sl
```

Set the file permissions on the installed Password Synchronizer shared library by executing the command:

```
chmod 755 /install_dir/SunOne/libsunpwsync*
```

For Solaris and HP-UX 64bit systems, create a subdirectory under *install_dir*/SunOne and copy the 64bit shared library `libsunpwsync_64.[sl|so]` to that directory, without the "_64" text in the library file name. Execute the following commands:

**On a HP 64bit host**

```
#cd /install_dir/SunOne
#mkdir pa20_64
#cp /install_dir/SunOne/libsunpwsync_64.sl
       /install_dir/SunOne/pa20_64/libsunpwsync.sl
```

**On a Solaris 64bit host**

```
#cd  /install_dir/SunOne
#mkdir 64
#cp  /install_dir/SunOne/libsunpwsync_64.so
       /install_dir/SunOne/64/libsunpwsync.so
```

## Configure the Sun ONE Directory Server Password Synchronization Plug-in

The configuration file /*install_directory*/SunOne/sunpwsync.props contains properties in the format *property_name=property_value*, with each property placed on a single line.

Specify the following properties (property names are case sensitive):

**jvmPath**
The path to the Java Virtual Machine executable (`java.exe`), used to run the Java layer (for example, C:\*install_directory*\-jvm\jre\bin).

**jvmClassPath**
The Java classpath of the Java layer. Contains the path to the JAR files needed by the Password Synchronization Plug-in and the Password Store. .

**syncClassName**
The custom synchronization class that implements the **com.ibm.di.plugin.pwsync.IPasswordSynchronizer** interface and stores the passwords.

**serverPort**
The port on which the C and Java layers communicate.

**logFile**
The C layer log file, used to log synchronization failures.

**syncBase**
The branch (base) of the directory tree where the Sun ONE Directory Server Password Synchronization Plug-in listens for password updates.

**javaLogFile**
The log file of the Java layer.

**checkRepository**
Specify **true** or **false** to correspondingly turn on or off the check for repository availability. When this option is turned on and the repository is not available, the Password Synchronization Plug-in cancels the password update in Sun ONE Directory Server.

The following is a (UNIX) sample of the Sun ONE Directory Server Password Synchronization Plug-in configuration file, `sunpwsync.props`:

```
jvmPath=install_dir/jvm/jre/bin
jvmClassPath=install_dir/SunOne:install_dir/SunOne
   /jars/proxy.jar:install_dir/SunOne/jars/ibmjms.jar:install_dir
   /SunOne/jars/idipwcrypto.jar:install_dir/SunOne/jars/idipwstore.jar:
   install_dir/SunOne/jars/auibase.jar:install_dirSunOne/
   jars/MQeBase.jar:install_dir/SunOne/jars/mqeconfig.jar:install_dir
   /SunOne/jars/MQeJMS.jar:install_dir/SunOne/jars/mqepwstore.jar:
   install_dir/SunOne/jars/MQeSecurity.jar:install_dir/
   SunOne/jars/stopproxy.jar:install_dir/SunOne/jars/tdipwflt.jar:
   install_dir/IDS/jars/MQeJMX.jar
syncClassName=com.ibm.di.plugin.pwsync.LDAPPasswordSynchronizer
serverPort=19013
logFile=install_dir/SunOne/sun_pw_sync.log
syncBase=dc=ibm,dc=com
javaLogFile=install_dir/SunOne/sun_pws_java.log
checkRepository=true
```

**Note:** *install_dir* is a variable indicating the installation directory. Do not copy and paste this example without first replacing variables with accurate data.

## Configuration

### Turning the Debug mode on or off

In order to turn the Sun ONE Directory Server Password Synchronization Plug-in DEBUG mode on, the lines that register the C plug-in in `install_directory/Servers/slapd-hostname/config/dse.ldif` must be modified. Before modifying these lines make sure that the Sun ONE Directory Server is shut down.

To turn DEBUG on, change the two lines that register the plug-in initialization functions:

```
nsslapd-pluginInitfunc: PWSyncInitPreOpDebug
...
nsslapd-pluginInitfunc: PWSyncInitPostOpDebug
```

To turn DEBUG off, restore these two lines back to their default values:

```
nsslapd-pluginInitfunc: PWSyncInitPreOp
...
nsslapd-pluginInitfunc: PWSyncInitPostOp
```

**Note:** By default DEBUG is turned off.

### Enabling Sun ONE Directory Server logging for plug-ins

Do the following to enable Sun ONE Directory Server logging for plug-ins:

1. On the Directory Server Console, select the **Configuration** tab.
2. In the navigation tree, expand the Logs folder and select the **Error Log** icon. The error log configuration attributes are displayed in the pane on the right side of the screen.
3. To enable error logging, select the **Enable Logging** check box (error logging is enabled by default).
4. Select **Plug-ins** in the Log Level list box and click **Save**.

### Integrating the Password Store

**Integrating with the LDAP Password Store**

To integrate the Sun ONE Directory Server Password Synchronization Plug-in with the LDAP Password Store, do the following:

1. In the Sun ONE Directory Server Password Synchronization Plug-in config file (`sunpwsync.props`), place the following value for the `syncClassName` property:

   ```
   syncClassName=com.ibm.di.plugin.pwsync.LDAPPasswordSynchronizer
   ```

   The class **com.ibm.di.plugin.pwsync.LDAPPasswordSynchronizer** is included in the `proxy.jar` file included in the Sun ONE Directory Server Password Synchronization Plug-in package.
2. Follow the configuration instructions of the LDAP Password Store (see Chapter 8, "LDAP Password Store," on page 73).
3. Place the LDAP Password Store configuration file in the folder where you installed the Sun ONE Directory Server Password Synchronization Plug-in (C:\*install_directory* for Windows or /usr/*install_directory* for UNIX and Linux).

#### Integrating with the MQe Password Store

To integrate the Sun ONE Directory Server Password Synchronization Plug-in with the MQe Password Store, do the following:

1. In the Sun ONE Directory Server Password Synchronization Plug-in config file (**sunpwsync.props**), place the following value for the `syncClassName` property:

   ```
   syncClassName=com.ibm.di.plugin.mqe.store.MQePasswordStore
   ```

   The class **com.ibm.di.plugin.mqe.store.MQePasswordStore** is included in the `mqepwstore.jar` file shipped with the MQe Password Store.
2. Follow the configuration instructions of the MQe Password Store (see Chapter 9, "MQ Everyplace Password Store," on page 85).
3. Place the configuration file of the MQe Password Store in the folder where you installed the Sun ONE Directory Server Password Synchronization Plug-in (`C:\`*`install_directory`* for Windows or `/usr/`*`install_directory`* for UNIX and Linux).

**Note:** You can change the Password Store used by the Sun ONE Directory Server Password Synchronization Plug-in. If the corresponding Password Store is installed and configured, do the following:

1. Shut down the Sun ONE Directory Server.
2. Edit the Sun ONE Directory Server Password Synchronization Plug-in configuration file and set the `syncClassName` accordingly.
3. Start the Sun ONE Directory Server again.

## Using the Password Synchronizer

Two of the configuration properties of the Sun ONE Directory Server Password Synchronizer are of particular interest and directly affect the password synchronization logic:

**syncBase**
: This property enables restricting the part of the directory tree where passwords are intercepted. The value specified is the LDAP distinguished name (dn) of the root of the tree whose entry' passwords you want to intercept. Specifying ″o=ibm,c=us″, for example, results in intercepting password update ″cn=Kyle Nguyen,ou=Austin,o=IBM,c=US″ and skipping the password update ″cn=Henry Nguyen,o=SomeOtherCompany,c=US″. Setting no value to this property results in the interception of password updates in the whole directory tree.

**checkRepository**

This property enables turning on or off the functionality that checks for availability of the Password Storage.

When this property is set to **true**, the Password Synchronizer first checks whether the Password Storage is available. If it is available, the password is changed in the directory, then the password is sent to the Password Storage. If the check indicates that the storage is not available, the LDAP operation (a part of which is the password update) is rejected on the Sun ONE Directory Server.

When the checkRepository property is set to **false**, the Password Synchronizer performs no checks for storage availability. The password update is performed in the directory first, then an attempt is made to store it in the Password Storage. If the password cannot be stored, a message is logged in the log file (pointed to by the `logFile` property) to indicate that password synchronization for this user failed.

**Note:** The check for availability of the Password Storage works with all Password Store components.

# Chapter 5. IBM Directory Server Password Synchronizer

The chapter describes the configuration and operation of the IBM Tivoli Directory Integrator Directory Server Password Synchronizer.

This chapter contains the following sections:

## Overview

The IBM Tivoli Directory Server Password Synchronizer intercepts changes to LDAP passwords in IBM Tivoli Directory Server.

Passwords in IBM Tivoli Directory Server are stored in the `userPassword` LDAP attribute. The Password Synchronizer intercepts updates of the `userPassword` LDAP attribute.

The IBM Tivoli Directory Server Password Synchronizer intercepts modifications of the `userPassword` attribute of entries of any object class.

Password updates are intercepted for the following types of entry modifications:

- When a new entry is added in the directory and the entry contains the `userPassword` attribute.
- When an existing entry is modified and one of the modified attributes is the `userPassword` attribute. This includes the following cases:
  - The `userPassword` attribute is added (for example, the entry did not have a `userPassword` attribute before)
  - The `userPassword` attribute is modified (for example, the entry had this attribute and its value is now changed)
  - The `userPassword` attribute is deleted from the entry

**Note:** Deletion of entries (users) is not intercepted by the IBM Tivoli Directory Server Password Synchronizer even when the entry contains the `userPassword` attribute.

**Note:** The `userPassword` attribute in IBM Tivoli Directory Server is multiple-valued. Users can have several passwords. The IBM Tivoli Directory Server Password Synchronizer intercepts and reports any change of any of the password values.

## Supported platforms

The IBM Tivoli Directory Server Password Synchronizer is available for the IBM Tivoli Directory Server on the following platforms and for the following versions:

- Windows 2000 Professional (IA32), IDS 5.2 FP4 (32 bit), IDS 6.0 FP1 (32 bit)
- Windows 2000 Server (IA32), IDS 5.2 FP4 (32 bit), IDS 6.0 FP1 (32 bit)
- Windows 2000 Advanced Server (IA32), IDS 5.2 FP4 (32 bit), IDS 6.0 FP1 (32 bit)
- Windows 2003 Standard Edition (IA32), IDS 5.2 FP4 (32 bit), IDS 6.0 FP1 (32 bit)
- Windows 2003 Enterprise Edition (IA32), IDS 5.2 FP4 (32 bit), IDS 6.0 FP1 (32 bit)
- AIX 5.2 (5200-80) (64 bit), IDS 5.2 FP4 (64 bit), IDS 6.0 FP1 (64 bit)
- AIX 5.3 (5300-03) (64 bit), IDS 5.2 FP4 (64 bit), IDS 6.0 FP1 (64 bit)
- Solaris 9 (32 bit and 64 bit), IDS 5.2 FP4 (32 bit), IDS 6.0 FP1 (32 bit)
- Solaris 10 (64 bit), IDS 6.0 FP1 (32 bit)
- HP-UX 11iv2 (11.23) (32 bit and 64 bit), IDS 5.2 FP4 (32 bit), IDS 6.0 FP1 (32 bit)
- RHEL AS 3.0 (IA32), IDS 5.2 FP4 (32 bit), IDS 6.0 FP1 (32 bit)
- RHEL AS 4.0 (IA32), IDS 6.0 FP1 (32 bit)
- SLES 8/UL 1.0 (IA32), IDS 5.2 FP4 (32 bit), IDS 6.0 FP1 (32 bit)
- SLES 9 (IA32), IDS 6.0 FP1 (32 bit)
- RedFlag Advanced Server 4.1 (IA32), IDS 6.0 FP1 (32 bit)

## Deployment

### Deploying Directory Server Password Synchronizer on Windows

#### Included Files

The following files are included with the distribution package:

**idspwsync.dll**
Contains the implementation which interfaces with IBM Directory Server account password change function for 32 bit Windows environments.

**idspwsync.props**
Configuration file required when using the LDAP Password Store.

**idicryptokeys.bat**
Utility batch file that creates test/demo keystore file.

**mqeconfig.props**
Configuration file required when using the MQe Password Store.

**JAR Files**
Contains the Password Synchronizer functionality that interfaces with the LDAP and MQe Password Stores: `proxy.jar`, `stopproxy.jar`, `mqepwstore.jar`, `idipwstore.jar`, `idipwcrypto.jar`, `mqeconfig.jar`, `auibase.jar`, `tdipwflt.jar`, `ibmjms.jar`, `MQeBase.jar`, `MQeJMS.jar`, `MQeSecurity.jar`, `MQeJMX.jar`

**LDIF Files**
Contain the LDAP schema definition required for the specific Directory Server used as the LDAP Password Store: `ibm-diPersonSchemaForAD.ldif`, `ibm-diPersonForSunDS.ldif` , `ibm-diPerson_oc.ldif`, `ibm-diPerson_z.ldif`.

The installation creates configuration files and places all other required files in the appropriate directory structure. The .jar files are located in `install-dir\jvm\jre\lib\ext\` where *install-dir*is the chosen installation directory. The .bat, .dll and .props files are located in `install-dir\IDS`.

## Registering the IBM Directory Server Password Synchronizer plug-ins within IBM Directory Server

To register the plug-in, edit the IBM Directory Server configuration file `ids_dir\etc\ibmslapd.conf`.

Find the section `dn: cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas, cn=Configuration` and add the following (as one line):

```
ibm-slapdPlugin: preoperation "install_dir\pwsync.dll" PWSyncInit
       "install_dir\idspwconfigWin.props"
```

**Note:** On IDS 5.2, the `ibmslapd.conf` configuration file will be located under the IDS home directory. For example, `ids_dir\etc\ibmslapd.conf`, where *ids_dir* is `/opt/IBM/ldap`. On IDS 6.0, the `ibmslapd.conf` configuration file is located under the database's instance directory. For example, `ids_dir\etc\ibmslapd.conf`, where *ids_dir* is `/export/home/idsldap/idsslapd-idsldap`.

## Configuring the IBM Directory Server Password Synchronizer

The configuration file `C:\install_dir\idspwconfigWin.props` contains properties in the format *property_name=property_value*, with each property placed on a single line.

Specify the following properties (property names are case sensitive):

**jvmPath**
: The path to the Java Virtual Machine executable (`java.exe`), used to run the Java layer (for example, `/install_dir/jvm/jre/bin`).

**jvmClassPath**
: The Java classpath of the Java layer. Contains the path to the JAR files needed by the Password Synchronizer and the Password Store.

**syncClassName**
: The custom synchronization class that implements the **com.ibm.di.plugin.pwsync.IPasswordSynchronizer** interface and stores the passwords.

**serverPort**
: The port on which the C and Java layers communicate.

**logFile**
: The C layer log file, used to log synchronization failures.

**syncBase**
: The branch (base) of the directory tree where the IBM Directory Server Password Synchronizer listens for password updates.

**javaLogFile**
: The log file of the Java layer.

**checkRepository**
: Specify **true** or **false** to correspondingly turn on or off the check for repository availability. When this option is turned on and the repository is not available, the Password Synchronizer cancels the password update in IBM Directory Server.

The following is a (Windows) sample of the IBM Directory Server Password Synchronizer configuration file, `idspwconfig.props`:

```
jvmPath=install_dir\jvm\jre\bin
jvmClassPath=install_dir\IDS;install_dir\IDS\jars\
   proxy.jar;install_dir\IDS\jars\ibmjms.jar;install_dir\
   IDS\jars\idipwcrypto.jar;install_dir\IDS\jars\idipwstore.jar;
   install_dir\IDS\jars\auibase.jar;install_dir\
   IDS\jars\MQeBase.jar;install_dir\IDS\jars\mqeconfig.jar;
   install_dir\IDS\jars\MQeJMS.jar;install_dir\
   IDS\jars\mqepwstore.jar;install_dir\IDS\jars\MQeSecurity.jar;
   install_dir\IDS\jars\stopproxy.jar;install_dir\
   IDS\jars\tdipwflt.jar;install_dir\IDS\jars\MQeJMX.jar
syncClassName=com.ibm.di.plugin.pwsync.LDAPPasswordSynchronizer
serverPort=19012
logFile=install_dir\IDS\ids_pw_sync.log
syncBase=O=IBM,C=US
javaLogFile=install_dir\IDS\ids_pws_java.log
checkRepository=true
```

**Note:** *`install_dir`* is a variable. Do not copy and paste this example without first replacing variables with accurate data.

## Deploying IBM Directory Server Password Synchronizer on UNIX and Linux

### Included Files

The following files are included in the distribution package:

**libidspwsync.so**
Contains the implementation which interfaces with IBM Directory Server account password change function for 32 bit Linux and Solaris UNIX environments.

**libidspwsync.a.so**
Contains the implementation which interfaces with IBM Directory Server account password change function for 32 bit AIX UNIX environments.

**libidspwsync_64.a.so**
Contains the implementation which interfaces with IBM Directory Server account password change function for 64 bit AIX UNIX environments.

**libidspwsync.sl**
Contains the implementation which interfaces with IBM Directory Server account password change function for 32 bit HP-UX UNIX environments.

**idspwconfig.props**
Configuration file required when using the LDAP Password Store.

**idicryptokeys.sh**
Utility script file that creates the test (demo) keystore file.

**mqeconfig.props**

**Configuration file required when using the MQe Password Store.**

**JAR Files**
Contain the Password Synchronizer functionality that interfaces with the LDAP and MQe Password Stores: `proxy.jar`, `stopproxy.jar`, `mqepwstore.jar`, `idipwstore.jar`, `idipwcrypto.jar`, `mqeconfig.jar`, `auibase.jar`, `tdipwflt.jar`, `ibmjms.jar`, `MQeBase.jar`, `MQeJMS.jar`, `MQeSecurity.jar`, `MQeJMX.jar`

**LDIF Files**
: Contain the LDAP schema definition required for the specific Directory Server used as the LDAP Password Store: `ibm-diPersonSchemaForAD.ldif`, `ibm-diPersonForSunDS.ldif` , `ibm-diPerson_oc.ldif`, `ibm-diPerson_z.ldif`.

The installation creates a configuration file and places all the other required files in the appropriate directory structures. The .jar files are located in *install_dir*/install_dir/jvm/jre/lib/ext/ where *install_dir* is the chosen installation directory. The .so/.a.so/.sl, .sh and .props files are located in *install_dir*/IDS.

## Registering the IBM Directory Server Password Synchronizer plug-in within IBM Directory Server

To register the plug-in, edit the IBM Directory Server configuration file *ids_dir*\etc\ibmslapd.conf.

Find the section `dn: cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas, cn=Configuration` and add the following (as one line):

**Win32**
```
ibm-slapdPlugin: preoperation "install_dir\IDS\pwsync.dll"
   PWSyncInit "install_dir\idspwconfig.props"
```

**AIX64**
```
ibm-slapdPlugin: preoperation "install_dir/IDS/libidspwsync_64.a.so
   PWSyncInit "install_dir/IDS/idspwconfig.props"
```

**Linux32**
```
ibm-slapdPlugin: preoperation "install_dirIDS/libidspwsync.so
   PWSyncInit
       "install_dir/IDS/idspwconfig.props"
```

**Note:** For IDS 5.2, the location of the `ibmslapd.conf` configuration file will be under the IDS home directory. For example, *ids_dir*\etc\ibmslapd.conf, where *ids_dir* is /opt/IBM/ldap. On IDS 6.0, the `ibmslapd.conf` configuration file is located under the database's instance directory. For example, *ids_dir*\etc\ibmslapd.conf, where *ids_dir* is /export/home/idsldap/idsslapd-idsldap.

## Configuring the IBM Directory Server Password Synchronizer

The configuration file /usr/*install_dir*/idspwconfigLnx.props contains properties in the format *property_name=property_value*, with each property placed on a single line.

Specify the following properties (property names are case sensitive):

**jvmPath**
: The path to the Java Virtual Machine executable (`java.exe`), used to run the Java layer (for example, C:\*install_dir*\jvm\jre\bin).

**jvmClassPath**
: The Java classpath of the Java layer. Contains the path to the JAR files needed by the Password Synchronizer and the Password Store.

**syncClassName**
: The custom synchronization class that implements the **com.ibm.di.plugin.pwsync.IPasswordSynchronizer** interface and stores the passwords.

**serverPort**
: The port on which the C and Java layers communicate.

**logFile**
The C layer log file, used to log synchronization failures.

**syncBase**
The branch (base) of the directory tree where the IBM Directory Server Password Synchronizer listens for password updates.

**javaLogFile**
The log file of the Java layer.

**checkRepository**
Specify **true** or **false** to correspondingly turn on or off the check for repository availability. When this option is turned on and the repository is not available, the Password Synchronizer cancels the password update in IBM Directory Server.

The following is a (UNIX) sample of the Directory Server Password Synchronizer configuration file, `idspwconfig.props`:

```
jvmPath=install_dir/jvm/jre/bin
jvmClassPath=install_dir/IDS:install_dir/IDS/jars/
   proxy.jar:install_dir/IDS/jars/ibmjms.jar:install_dir/
   IDS/jars/idipwcrypto.jar:install_dir/IDS/jars/idipwstore.jar:
   install_dir/IDS/jars/auibase.jar:install_dir/
   IDS/jars/MQeBase.jar:install_dir/IDS/jars/mqeconfig.jar:
   install_dir/IDS/jars/MQeJMS.jar:install_dir/
   IDS/jars/mqepwstore.jar:install_dir/IDS/jars/MQeSecurity.jar:
   install_dir/IDS/jars/stopproxy.jar:install_dir/
   IDS/jars/tdipwflt.jar:install_dir/IDS/jars/MQeJMX.jar
syncClassName=com.ibm.di.plugin.pwsync.LDAPPasswordSynchronizer
serverPort=19012
logFile=install_dir/IDS/ids_pw_sync.log
syncBase=O=IBM,C=US
javaLogFile=install_dir/IDS/ids_pws_java.log
checkRepository=true
```

**Note:** *install_dir* is a variable indicating the installation directory. Do not copy and paste this example without first replacing variables with accurate data.

## Configuration

### Turning the Debug mode on or off

In order to turn the IBM Directory Server Password Synchronizer DEBUG mode on, the lines that register the C plug-in in `install_dir/etc/ibmslapd.conf` must be modified.

Before modifying these lines make sure that the IBM Directory Server is shut down.

To turn DEBUG on, modify the following lines that register the plug-in initialization functions (the initialization routine `PWSyncInit` is replaced with `PWSyncInitDebug` ):

For Windows:

```
ibm-slapdPlugin: preoperation "C:/<install_dir>/pwsync.dll"
   PWSyncInitDebug "C:\install_dir\idspwconfigWin.props"
```

For UNIX and Linux:

```
ibm-slapdPlugin: preoperation install_dir/IDS/libidspwsync_64.so
   PWSyncInitDebug install_dir/IDS/idspwconfigLnx.props
```

To turn DEBUG off, restore these lines back to their default values:

For Windows:

```
ibm-slapdPlugin: preoperation "C:/<install_dir>/pwsync.dll"
   PWSyncInit "C:\install_dir\idspwconfigWin.props"
```

For UNIX and Linux:

```
ibm-slapdPlugin: preoperation install_dir/IDS/libidspwsync_64.so
   PWSyncInit install_dir/IDS/idspwconfigLnx.props"
```

**Note:** By default DEBUG is turned off.

## Integrating the Password Store

The following sections describe how to integrate the Password Store.

### Integrating with the LDAP Password Store

To integrate the IBM Directory Server Password Synchronizer with the LDAP Password Store, do the following:

1. In the IBM Directory Server Password Synchronizer config file, `idspwconfig.props`, place the following value for the `syncClassName` property:

   ```
   syncClassName=com.ibm.di.plugin.pwsync.LDAPPasswordSynchronizer
   ```

   The class `com.ibm.di.plugin.pwsync.LDAPPasswordSynchronizer` is included in the `proxy.jar` file, which is shipped with the LDAP Password Store.
2. Follow the configuration instructions of the LDAP Password Store. See Chapter 8, “LDAP Password Store,” on page 73.
3. Place the LDAP Password Store configuration file in the folder where you installed the IBM Directory Server Password Synchronizer.

For instructions on how to configure the properties file for the LDAP Password Store, see “Preparing the LDAP Password Store properties file (idipwsync.props)” on page 78.

### Integrating with the MQe Password Store

To integrate the IBM Directory Server Password Synchronizer with the MQe Password Store, do the following:

1. In the IBM Directory Server Password Synchronizer config file (**idspwconfigWin.props** or **idspwconfigLnx.props**), place the following value for the `syncClassName` property:

   ```
   syncClassName=com.ibm.di.plugin.mqe.store.MQePasswordStore
   ```

   The class `com.ibm.di.plugin.mqe.store.MQePasswordStore` is included in the `mqepwstore.jar` file shipped with the MQe Password Store.
2. Follow the configuration instructions of the MQe Password Store (see Chapter 9, “MQ Everyplace Password Store,” on page 85).
3. Place the configuration file of the MQe Password Store in the folder where you installed the IBM Directory Server Password Synchronizer (`C:\`*`install_dir`* for Windows or `/usr/`*`install_dir`* for UNIX and Linux).

**Note:** You can change the Password Store used by the IBM Directory Server Password Synchronizer. If the corresponding Password Store is installed and configured, do the following:

1. Shut down the IBM Directory Server.
2. Edit the IBM Directory Server Password Synchronizer configuration file and set the `syncClassName` accordingly.
3. Start the IBM Directory Server again.

### Stopping the Java layer (Java Proxy)

When the IBM Directory Server is shut down, the Java Layer of the IBM Directory Server Password Synchronizer is not automatically terminated. If you do not terminate the Java Layer explicitly, the IBM Directory Server Password Synchronizer does not start properly the next time the IBM Directory Server is activated.

For terminating the Java Layer you can use the **StopProxy** utility. It is a standalone Java application that sends a command to the IBM Directory Server Password Synchronizer Java layer requesting termination. You can run this utility manually or include it in IBM Directory Server shutdown scripts.

The **StopProxy** is distributed in the `proxy.jar` Java archive, which is placed in *`install_dir`*`/IDS/jars`. To use the **StopProxy** (entered as one line):

```
"install_dir/jvm/jre/bin/java"
      -DpropertiesFile=password_synchronizer_config_file
      -cp "install_dir/IDS/jars/proxy.jar"
      com.ibm.di.plugin.pwsync.StopProxy
      port [hostIP]
```

where:

- *password_synchronizer_config_file* is the fully qualified file system path of the password synchronizer configuration file (`idspwconfig.props`).
- *port* is a required parameter and represents the port used for communication between the C and Java layer (use the value of the serverPort configuration parameter).
- [*hostIP*] is an optional parameter that, when specified, represents the IP address of the machine where the IBM Directory Server Password Synchronizer is deployed (when omitted, a default value of **localhost** is assumed).

## Using the Password Synchronizer

Two of the configuration properties of the IBM Tivoli Directory Server Password Synchronizer are of particular interest and directly affect the password synchronization logic:

**syncBase**
: This property enables restricting the part of the directory tree where passwords are intercepted. The value specified is the LDAP distinguished name (dn) of the root of the tree whose entry passwords you want to intercept. Specifying ″o=ibm,c=us″, for example, results in intercepting the password update ″cn=Kyle Nguyen,ou=Austin,o=IBM,c=US″ and skipping password update of ″cn=Henry Nguyen,o=SomeOtherCompany,c=US″. Setting no value to this property results in the interception of password updates in the whole directory tree.

**checkRepository**
: This property enables turning on or off the functionality that checks for availability of the Password Storage.

When this property is set to *true*, the Password Synchronizer first checks whether the Password Storage is available. If it is available, the password is changed in the directory, then the password is sent to the Password Storage. If the check indicates that the storage is not available, the LDAP operation (a part of which is the password update) is rejected on the IBM Tivoli Directory Server.

When the **checkRepository** property is set to *false*, the Password Synchronizer performs no checks for storage availability. The password update is performed in the directory first, then an attempt is made to store it in the Password Storage. If the password cannot be stored, a message is logged in the log file (pointed to by the `logFile` property) to indicate that password synchronization for this user failed.

**Note:** The check for availability of the Password Storage works with all Password Store components.

## Stopping the Proxy Layer

The IBM Tivoli Directory Server Password Synchronizer consists of two layers: an IBM Tivoli Directory Server plug-in which is hooked into the Server, and a Java Proxy Layer. The plug-in intercepts password updates and sends them to the Proxy Layer. The Proxy Layer instantiates the Password Store component on startup and transmits all password updates received by the plug-in to the Password Store.

The Proxy Layer is started automatically by the IBM Director Server when the IBM Tivoli Directory Server starts. However, it is not stopped when the IBM Tivoli Directory Server stops. The Proxy Layer must be stopped explicitly when the IBM Tivoli Directory Server is shut down.

If you do not stop the Java Layer explicitly, the IBM Tivoli Directory Server Password Synchronizer does not start properly the next time the IBM Tivoli Directory Server is activated.

Use the **StopProxy** utility included in the IBM Tivoli Directory Server Password Synchronizer to stop the Java Layer. It is a standalone Java application and its usage is described in “Stopping the Java layer (Java Proxy)” on page 42.

# Chapter 6. Domino HTTP Password Synchronizer

This chapter describes the configuration and operation of the IBM Tivoli Directory Integrator Domino HTTP Password Synchronizer.

This chapter contains the following sections:
- "Overview"
- "Supported platforms"
- "Deployment and configuration" on page 46
- "Using the Password Synchronizer" on page 58

## Overview

The Domino HTTP Password Synchronizer intercepts changes of the Internet password (also known as HTTP password) for Notes users.

The following types of password changes are intercepted:

**Administrative password resets**
A user with the necessary rights (usually an administrator) changes his or another user's password without being prompted for the old password:
- The HTTP password is changed by editing the **Internet password** field of the user's Person document using the Lotus Domino Administrator client.
- The HTTP password is changed by editing the **Internet password** field of the user's Person document using the Web browser interface.

**Normal user password changes**
A user changes his own password and is prompted for the old password:
- A user changes his password from a Web browser using the **Change Password** form from the `domcfg.nsf` (Domino Web Server Configuration) database.
- A user changes his password from iNotes™.

## Supported platforms

The Domino HTTP Password Synchronizer is supported on the following platforms:
- Windows 2000 Server (IA32), Domino 6.5.4 and Domino 7.0
- Windows 2000 Advanced Server (IA32), Domino 6.5.4 and Domino 7.0
- Windows 2003 Standard Edition (IA32), Domino 6.5.4 and Domino 7.0
- Windows 2003 Enterprise Edition (IA32), Domino 6.5.4 and Domino 7.0
- AIX 5.2 (5200-08) (32 bit), Domino 6.5.4 and Domino 7.0
- AIX 5.3 (5300-03) (32 bit), Domino 6.5.4 and Domino 7.0
- Solaris 9 (64 bit), Domino 6.5.4 and Domino 7.0
- Solaris 10 (64 bit), Domino 6.5.4 and Domino 7.0
- SLES 8/UL 1.0 (IA32), Domino 6.5.4 and Domino 7.0

## Deployment and configuration

The Domino HTTP Password Synchronizer can be deployed in the following modes:

- Both administrative password resets and normal user password changes are intercepted.
- Only normal user password changes are intercepted.
- Only administrative password resets are intercepted.

For deployment and configuration of the Domino HTTP Password Synchronizer see the following sections.

## Deployment on a single Domino R6 Server

To install the Domino HTTP Password Synchronizer on Domino R6 run the installer on the machine where the Domino R6 Server is installed. The installer places all required files in the appropriate directory structures.

The following are file paths of Domino R6 Server directories:

- The Domino R6 Server Program Folder is referred to as *domino_program_directory* (for example, `C:\Lotus\Domino` on Windows, `/opt/lotus` on Linux- and UNIX-based platforms).
- The Domino R6 Server Data Folder is referred to as *domino_data_directory* (for example, `C:\Lotus\Domino\Data` on Windows, `/local/notesdata` on Linux- and UNIX-based platforms).
- The Domino R6 Server JVM Folder is referred to as *domino_jvm_directory* (for example, `C:\Lotus\Domino\jvm` on Windows, `/opt/lotus/notes/60000/linux/jvm` on Linux- and UNIX-based platforms).

### Included files (Windows)

*domino_data_directory*\**idipwsync_install_r6.nsf**
: This database contains the updated Person document forms and agent code of the Password Synchronizer.

*domino_data_directory*\**idipwsync.nsf**
: A database used for temporary storage of intercepted passwords.

*domino_program_directory*\**idipwsync\\ididompwsync.props**
: Configuration file of the Password Synchronizer.

*domino_program_directory*\**idipwsync\\idipwsync.props**
: Configuration file required when using the LDAP Password Store.

*domino_program_directory*\**idipwsync\\jce**\*JAR Files*
: Contains the required JCE and JSSE JAR files: `ibmjcefw.jar`, `ibmjceprovider.jar`, `ibmpkcs11.jar`, `ibmpkcs.jar`, `local_policy.jar`, `US_export_policy.jar`, `ibmjsse.jar`.

*domino_program_directory*\**idipwsync\\idicryptokeys.bat**
: Utility batch file that creates test/demo keystore file.

*domino_program_directory*\**idipwsync\\mqeconfig.props**
: Configuration file required when using the MQe Password Store.

*domino_jvm_directory*\**lib\\ext \\JAR Files**
: Contains the Password Synchronizer functionality that interfaces with the LDAP and MQe Password Stores: `proxy.jar`, `stopproxy.jar`, `mqepwstore.jar`, `idipwstore.jar`, `idipwcrypto.jar`, `mqeconfig.jar`, `ibmjms.jar`, `MQeBase.jar`, `MQeJMS.jar`, `MQeSecurity.jar`, `MQeJMX.jar`.

*domino_program_directory***idipwsync\LDIF Files**
: Contain the LDAP schema definition required for the specific Directory Server used as the LDAP Password Store: `ibm-diPersonSchemaForAD.ldif`, `ibm-diPersonForSunDS.ldif` , `ibm-diPerson_oc.ldif`, `ibm-diPerson_z.ldif`.

## Included files (Linux and UNIX)

*domino_data_directory*/ **idipwsync_install_r6.nsf**
: This database contains the updated Person document forms and agent code of the Password Synchronizer.

*domino_data_directory*/**idipwsync.nsf**
: A database used for temporary storage of intercepted passwords.

*domino_data_directory*/**idipwsync/ididompwsync.props**
: Configuration file of the Password Synchronizer.

*domino_data_directory*/**idipwsync/idipwsync.props**
: Configuration file required when using the LDAP Password Store.

*domino_data_directory*/**idipwsync/jce/***JAR Files*
: Contains the required JCE and JSSE JAR files: `ibmjcefw.jar`, `ibmjceprovider.jar`, `ibmpkcs11.jar`, `ibmpkcs.jar`, `local_policy.jar`, `US_export_policy.jar`, `imbjsse.jar`.

*domino_data_directory*/**idipwsync/idicryptokeys.sh**
: Utility script file that creates test/demo keystore file.

*domino_data_directory*/**idipwsync/mqeconfig.props**
: Configuration file required when using the MQe Password Store.

*domino_jvm_directory*/**lib/ext /JAR Files**
: Contains the Password Synchronizer functionality that interfaces with the LDAP and MQe Password Stores: `proxy.jar`, `stopproxy.jar`, `mqepwstore.jar`, `idipwstore.jar`, `idipwcrypto.jar`, `mqeconfig.jar`, `ibmjms.jar`, `MQeBase.jar`, `MQeJMS.jar`, `MQeSecurity.jar`, `MQeJMX.jar`.

## Setup

Do the following to set up Domino R6:

1. Restart the Domino Server.
2. Sign `idipwsync_install_r6.nsf` and `idipwsync.nsf` with Server ID:
   a. Start Lotus Domino Administrator.
   b. Select **Files**.
   c. Right-click on the **idipwsync_install_r6** database and select **Sign**.
   d. In **Sign Database**, under **Which ID do you want to use?**, select **Active Server's ID**.
   e. Right-click on the **idipwsync** database and select **Sign**.
   f. In **Sign Database**, under **Which ID do you want to use?**, select **Active Server's ID**.
   g. Click **OK**.
3. Update the design of `pubnames.ntf` template:
   a. Start Lotus Domino Designer®.
   b. Open the following:
      1) Open **idipwsync_install_r6.nsf** database.
      2) Open `pubnames.ntf` template.

c. Copy Agents:
   1) In `idipwsync_install_r6.nsf`, select **Shared Code/Agents**.
   2) Select both **IDIPWSyncClientAgent** and **IDIPWSyncWebAgent** (press the **Ctrl** key while clicking the two agents).
   3) Right-click on the selected agents and select **Copy**.
   4) In `pubnames.ntf`, select **Shared Code/Agents**.
   5) Go to **Edit->Paste** to paste the two agents.

   If the Person form has not been modified with user-customized logic and the Domino Server version is 6.5.4, the Person form from the Password Synchronizer is used.
d. Rename the Person form in `pubnames.ntf`:
   1) In `pubnames.ntf` select **Forms**.
   2) Open the **Person** form.
   3) Go to **Design->Form Properties**.
   4) Edit the **Name** field. Change the name to **original_Person** (or other name of your choice, other than **Person**).
   5) Save the form.
   6) Close the form.
e. Copy the Person form:
   1) In `idipwsync_install_r6.nsf` select **Forms**.
   2) Right-click on the **Person** form and select **Copy**.
   3) In `pubnames.ntf` select **Forms**.
   4) Go to **Edit->Paste** to paste the form.

   If the Person form has been modified with user-customized logic that needs to be kept, or if the Domino Server version is different from 6.5.4, Password Synchronized source code for the Person form is copied manually.
f. Copy Person form source code:
   1) Copy **WebQuerySave** event code:
      a) In `idipwsync_install_r6.nsf`, select **Forms**.
      b) Open the **Person** form.
      c) Select the **WebQuerySave** event.
      d) Copy the lines starting with **REM {start of IDI Password Synchronizer code};** and ending with **REM {end of IDI Password Synchronizer code};**
      e) In `pubnames.ntf` select **Forms**.
      f) Open the **Person** form.
      g) Select the **WebQuerySave** event.
      h) Paste the copied source code. Make sure the pasted code appears before any other code in this event.
      i) Save the form.
   2) Copy **QuerySave** event code:
      a) In `idipwsync_install_r6.nsf`, select **Forms**.
      b) Open the **Person** form.
      c) Select the **QuerySave** event.
      d) Copy the lines starting with **'start of Password Synchronizer code** and ending with **'end of Password Synchronizer code**.

e) In `pubnames.ntf`, select **Forms**.

f) Open the **Person** form.

g) Select the **QuerySave** event.

h) Paste the copied source code. Make sure the pasted code appears just before the end of the **Querysave** procedure.

i) Save the form.

3) Copy **SyncPass** event code:

a) In `idipwsync_install_r6.nsf`, select **Forms**.

b) Open the **Person** form.

c) Select the **SyncPass** event.

d) Copy all code for the **SyncPass** function.

e) In `pubnames.ntf`, select **Forms**.

f) Open the **Person** form.

g) Select the **QuerySave** event.

h) Paste the copied source code. Make sure the pasted code appears after all code in the event. A new event named **SyncPass** is created immediately, and the pasted code is transferred there.

i) Save the form.

If the **$PersonInheritableSchema** subform has not been modified with user-customized logic and the Domino Server version is 6.5.4, the **$PersonInheritableSchema subform** from the Password Synchronizer is used.

g. Rename the **$PersonInheritableSchema** subform in `pubnames.ntf`:

1) In `pubnames.ntf`, select **Shared Code/Subforms**.

2) Open the **$PersonInheritableSchema** subform.

3) Go to **Design->Subform Properties**.

4) Edit the **Name** field. Change the name to **original_$PersonInheritableSchema** (or other name of your choice other than **$PersonInheritableSchema**).

5) Save the form.

6) Close the form.

h. Copy the **$PersonInheritableSchema** form:

1) In `idipwsync_install_r6.nsf`, select **Shared Code/Subforms**.

2) Right-click on the **$PersonInheritableSchema** form and select **Copy**.

3) In `pubnames.ntf`, select **Shared Code/Subforms**.

4) Go to **Edit->Paste** to paste the form.

If the **$PersonInheritableSchema** subform has been modified with user-customized logic that needs to be kept, or if the Domino Server version is different from 6.5.4, Password Synchronized source code is copied manually.

i. Copy **$PersonInheritableSchema** subform source code:

1) Copy **HTTPPassword** field code:

a) In `idipwsync_install_r6.nsf`, select **Shared Code/Subforms**.

b) Open the **$PersonInheritableSchema** subform.

c) Select the **HTTPPassword** field (near the bottom of the form).

d) Select the **Input Translation** event.

   e) Copy the lines starting with **REM {start of IDI Password Synchronizer code};** and ending with **REM {end of IDI Password Synchronizer code};**
   f) In `pubnames.ntf`, select **Shared Code/Subforms**.
   g) Open the **$PersonInheritableSchema** form.
   h) Select the **HTTPPassword** field.
   i) Select the **Input Translation** event.
   j) Paste the copied source code. Make sure the pasted code appears before any other code in this event.
   k) Save the form.

2) Copy **FullName** field code:
   a) In `idipwsync_install_r6.nsf`, select **Shared Code/Subforms**.
   b) Open the **$PersonInheritableSchema** subform.
   c) Select the **FullName** field (near the bottom of the form).
   d) Select the **Input Validation** event.
   e) Copy the lines starting with **REM {start of IDI Password Synchronizer code};** and ending with **REM {end of IDI Password Synchronizer code};**
   f) In `pubnames.ntf`, select **Shared Code/Subforms**.
   g) Open the **$PersonInheritableSchema** form.
   h) Select the **FullName** field.
   i) Select the **Input Validation** event.
   j) Paste the copied source code before any other code in this event.
   k) Save the form.

4. Refresh the design of the `names.nsf` database:
   a. In Lotus Domino Administrator select **Files**.
   b. Select `names.nsf` database.
   c. Go to **File->Database->Refresh Design**.
   d. Select the name of your server from the **With Design from Server** list.
   e. Click **OK**.
   f. Click **Yes** to continue.
5. Update the design of `admin4.ntf` template:
   a. Copy the file `proxy.jar` from the *`domino_jvm_directory`*`/lib/ext` folder on the Domino Server to the `Lotus\Notes\jvm\lib\ext` folder on the machine where Lotus Domino Designer is installed.
   b. In Lotus Domino Designer open `admin4.ntf` template database and `idipwsync_install_r6.nsf` database.
   c. Copy the **IDIPWSyncAdminRequestAgent**:
      1) In `idipwsync_install_r6.nsf`, select **Shared Code/Agents**.
      2) Select the **IDIPWSyncAdminRequestAgent**.
      3) Right-click on the selected agent and select **Copy**.
      4) In `admin4.ntf` select **Shared Code/Agents**.
      5) Go to **Edit->Paste** to paste the agent.
   d. Configure the **IDIPWSyncAdminRequestAgent**:
      1) Open the **IDIPWSyncAdminRequestAgent**.
      2) Go to **Edit->Properties**.

        3) Click **Edit settings** from the Runtime section of the Agent dialog box.
        4) In the **Run on** field select the name of the current Domino server.
        5) Click **OK**.
        6) Close the agent dialog box.
        7) Go to **File->Save** to save the new agent settings.
6. Refresh the design of the `admin4.nsf` database:
    a. In Lotus Domino Administrator select **Files**.
    b. Select `admin4.nsf` database.
    c. Go to **File->Database->Refresh Design**.
    d. Select the name of your server from the **With Design from Server** list.
    e. Click **OK**.
    f. Click **Yes** to proceed.
7. Setup secret key encryption infrastructure.

    Secret key encryption is used to protect passwords in the time slice in which they are temporarily stored in a database on the Domino Server.

    a. Generate a secret key:
        1) In Lotus Domino Administrator, go to **File->Security->User Security**.
        2) Select **Notes Data/Documents** from the left navigation panel.
        3) Click **New Secret Key**.
        4) Enter **IdiPwSync** as secret key name and click **OK**.
        5) Click **Other Actions** and select **Export secret key**.
        6) Enter a password to protect the exported secret key.

            **Note:** This step is optional but highly recommended.
        7) Save the key in a file named `idipwsync.key`.
        8) Click **Close** in the **User Security** screen.

    b. Import the secret key in the Domino Server ID file:
        1) Stop the Domino Server.
        2) In Lotus Domino Administrator, go to **File->Security->Switch ID**.
        3) Open the `server.id` file for the Domino Server. To do so you must use either a Lotus Domino Administrator installed on the Domino Server machine, or copy the `server.id` file to the machine where Lotus Domino Administrator is installed. The `server.id` file is usually placed in `domino_data_directory`.
        4) Go to **File->Security->User Security**.
        5) Select **Notes Data/Documents** from the left navigation panel.
        6) Click **Other Actions** and select **Import secret key**.
        7) Open the `idipwsync.key` file.
        8) If the file is password protected, enter the password that was created when you exported the secret key (see ″Generate a secret key,″ previous).
        9) Click **Accept** to import the secret key.
        10) Click **Close** in the **User Security** screen.
        11) Go to **File->Security->Switch ID** and switch back to the administrator ID file.

12) If you edited a copy of the `server.id` file, copy it over the original `server.id` file in *domino_data_directory* (you might want to backup the original `server.id` before overwriting it with the new one).

13) Start the Domino Server.

c. Import the secret key in the ID files of all Administrators or users that can edit Person documents and change http passwords. For each of these Administrators or users, do the following:

1) In Lotus Domino Administrator, go to **File->Security->Switch ID**.

2) Open the ID file of the Administrator or user.

3) Go to **File->Security->User Security**.

4) Select **Notes Data/Documents** from the left navigation panel.

5) Click **Other Actions** and select **Import secret key**.

6) Open the `idipwsync.key` file.

7) If the file is password protected, enter the password that was created when you exported the secret key (see the steps for generating a secret key in part (a) above).

8) Click **Accept** to import the secret key.

9) Click **Close** in the **User Security** screen.

**Note:** An Administrator or user whose ID file does not contain the secret encryption key is not allowed to change the HTTP Password field of Person documents.

8. Setup port encryption (optional).

Port encryption encrypts the communication between Lotus Domino Administrator and the Domino Server, bringing an additional layer of security to the network communication.

**Note:** Port encryption is recommended but not required. Prior to being sent over the network, the password is encrypted with the secret key, whether port encryption is used or not.

Two options are available:

- Setup the Domino Server to encrypt communication ports. This is easier to set up (the Server settings only are configured) but it affects the communication with all clients, including regular users using Lotus Notes clients.
- Setup the Lotus Domino Administrator clients to encrypt communication ports. This requires configuration of each Lotus Domino Administrator client that is used, but does affect other Notes clients if encryption is not necessary for them. Do this as follows:

a. Encrypt Domino Server communication ports:

1) In Lotus Domino Administrator select **Configuration**.

2) Select **Server/Server Ports** from the right-side panel.

3) For each communication port in use, select the port in the **Communication ports** list and check the **Encrypt network data** option.

4) Click **OK**.

5) Restart the Domino Server for changes to take effect.

b. Encrypt Lotus Domino Administrator communication ports:

Do the following for each Lotus Domino Administrator client that can be used for password changes:

1) In Lotus Domino Administrator go to **File->Preferences->User Preference**.
2) Select **Ports** from the left navigation panel.
3) For each communication port in use, select the port in the **Communication ports** list and check the **Encrypt network data** option.
4) Click **OK**.
5) Restart Lotus Domino Administrator for changes to take effect.

9. Setup SSL for Domino HTTP Server.

   SSL is necessary to secure the communication between the Web browser and the Domino HTTP Server. If SSL is not set up, the password is transferred over the network in plain text.

   Consult the Lotus Domino Administrator help documentation for more information about setting up SSL ("Setting up SSL on a Domino server" is a recommended article).

10. **For Domino 6.5.4 ONLY**, configure Domino JVM security provider:

    Domino JVM security settings must be updated to enable JCE. JCE is used by the IBM Tivoli Directory Integrator Password Store components to encrypt and decrypt password data. Do the following to enable JCE in the Domino Server JVM:

    a. The following files must be copied from the installed JCE staging JAR file directory to the *domino_jvm_dir*/`lib/ext` folder:

       **Note:** For Windows installs, the source JCE folder is *domino_ program_dir*/`idipwsync/jce` and for UNIX installs the source JCE folder is *domino_ data_dir*/`idipwsync/jce`.

       - `ibmjcefw.jar`
       - `ibmjceprovider.jar`
       - `ibmpkcs.jar`
       - `ibmpkcs11.jar`
       - `local_policy.jar`
       - `US_export_policy.jar`
       - `ibmjsse.jar`

    b. Remove the `ibmjcaprovider.jar` file from *domino_jvm_directory*/`lib/ext` folder (if the file exists).

    c. Edit the file *domino_jvm_directory*/`lib/security/java.security`. If the following line exists:

       ```
       security.provider.x=com.ibm.crypto.provider.IBMJCA
       ```

       (where *x* represents a number, usually **2**), replace it with the following line:

       ```
       security.provider.x=com.ibm.crypto.provider.IBMJCE
       ```

       If an IBMJCA provider is not specified, add a new line for the IBMJCE provider:

       ```
       security.provider.x=com.ibm.crypto.provider.IBMJCE
       ```

       where *x* is one more than the value of the last security.provider entry.

11. Setup Domino Agents and Proxy Process:

The file ididompwsync.props contains the configuration for the **IDIPWSyncClientAgent**, **IDIPWSyncWebAgent**, **IDIPWSyncAdminRequestAgent** and the Proxy Process.

The following properties are available:

**proxy.storeClassName**
: The custom synchronization class that implements the **com.ibm.di.plugin.pwsync.IPasswordSynchronizer** interface. This class represents the Password Store implementation. The Proxy Process loads it on startup and invokes it through the **IPasswordSynchronizer** interface to store intercepted passwords

**proxy.port**
: The port on which the Communication Library and the Proxy Process communicate. This property is usually set to **19003**.

**proxy.syncStoreAccess**
: Specify **true** or **false** to correspondingly turn synchronization of the access to the Password Store on or off. Synchronizing Password Store access is useful in cases where the Password Store does not support multithreaded access or performs poorly when accessed by multiple threads (such as MQ Everyplace).

**proxy.logFileName**
: The log file of the Proxy Process.

**agent.client.logFileName**
: The log file of the Agent that is triggered when passwords are changed through the Lotus Domino Administrator.

**agent.web.logFileName**
: The log file used by the Agent that is triggered when passwords are changed through the Web browser.

**agent.adminp.logFileName**
: The log file used by the Agent that is triggered when passwords are changed through iNotes or through the **Change Password** Web form from the `domcfg.nsf` database.

**debug** Specify **true** or **false** to correspondingly turn debug information on or off.

The following is a sample **ididompwsync.props** file:

```
proxy.storeClassName=com.ibm.di.plugin.mqe.store.MQePasswordStore
proxy.port=19003
proxy.syncStoreAccess=true
proxy.logFileName=idipwsync/proxy.log
agent.client.logFileName=idipwsync/agent_client.log
agent.web.logFileName=idipwsync/agent_web.log
agent.adminp.logFileName=idipwsync/agent_adminp.log
debug=true
```

**Note:** There is an issue with the installer on some UNIX platforms for the Domino Password Synchronizer. If you have selected MQe as the password store, confirm that the **proxy.storeClassName** property has been set to `com.ibm.di.plugin.mqe.store.MQePasswordStore`. If not, manually update this property with the correct value for use of the MQe Password Store.

12. Configure the Domino Server to automatically start and stop the Proxy Process:

Open the file `domino_program_directory/notes.ini` and find the **ServerTasks** property. Add the following value at the end of the **ServerTasks** property:

```
runjava com.ibm.di.plugin.pwsync.domino.DominoProxy
```

The following is a sample **ServerTasks** property in `notes.ini`:

```
ServerTasks=Update,Replica,Router,AMgr,AdminP,CalConn,Sched,HTTP,
runjava com.ibm.di.plugin.pwsync.domino.DominoProxy
```

13. Configure Execution control list of Lotus Domino Administrator clients:

    Do the following for each Lotus Domino Administrator client that can be used for password changes:

    a. In Lotus Domino Administrator go to **File->Security->User Security**.
    b. Select **What Others Do/Using Workstation** in the left navigation panel.
    c. In the **When code is signed by** list, select the name of your Domino Server, for example **serverName/certifierName**. If the name of your Domino Server is missing, add it to this list.
    d. Under **Allow access to:**, check the **Current database** option.
    e. Under **Allow ability to:**, check the **Read other databases** and **Modify other databases** options.
    f. Click **OK**.
14. Configure Access Control:
    a. Create **IDIPWSync** group in Domino Directory:
       1) In Lotus Domino Administrator, select **People & Groups**.
       2) In the left navigation panel, select **Domino Directories/*your_domain*'s Directory/Groups** where *your_domain* is the name of the Lotus Domino domain.
       3) Click **Add Group**.
       4) Type **IDIPWSync** in **Group name**.
       5) In the **Members** field add all Administrators or users that can change passwords by editing Person documents.
    b. Grant permissions to execute Password Synchronizer Agents:
       1) In Lotus Domino Administrator, select **Configuration**.
       2) In the left navigation panel, select **Server/Current Server Document**.
       3) Select the **Security** page.
       4) In the field **Run unrestricted methods and operations**, add the **IDIPWSync** group.
       5) Click **Save & Close** to save the changes to the server document.
    c. Update Access Control List of the `idipwsync.nsf` database:
       1) In Lotus Domino Administrator, select **Files**.
       2) Select the `idipwsync.nsf` database.
       3) Select **Database/Manage ACL** from the right-side panel.
       4) Click **Add** and select the **IDIPWSync** group.
       5) Set **Access** to **Editor**.
       6) Set the following options under Attributes:
          - Check the **Delete Documents option**. The options **Create Documents**, **Read Public Documents**, and **Write Public Documents** must be checked as well. This is done automatically when **Editor** access is selected.

- Uncheck the options **Create private agents**, **Create personal folders/views**, **Create shared folders/views**, **Create LotusScript/Java agents**, **Replicate** or **copy documents**.

7) Select **Default** from the **Access Control List**.
8) Set **Access** to **No Access**.
9) Click **OK**.

**Note:** After the `idipwsync.nsf` database ACL is changed, it is no longer possible to change this ACL from the Domino Server. For security reasons, the most restrictive settings are used. If a new change of the ACL is necessary, the database must be opened locally and its ACL changed as needed.

15. Integration with the Password Store:
    a. Specifying or changing the Password Store (initially performed by the installer):

       The Password Store used with the Domino HTTP Password Synchronizer is specified through the **proxy.storeClassName** property in the **ididompwsync.props** configuration file. Valid values are:
       - **proxy.storeClassName= com.ibm.di.plugin.pwsync.LDAPPasswordSynchronizer** to use the LDAP Password Store.
       - **proxy.storeClassName= com.ibm.di.plugin.mqe.store.MQePasswordStore** to use the MQe Password Store.

    b. Deploying the Password Store files (performed by the installer):

       **Note:** On Linux- and UNIX-based platforms, install the Password Store with the Domino user (**notes** by default). This gives the necessary privileges to the Domino JVM to execute the Password Store.

       1) Jar files are placed in the *domino_jvm_directory*`/lib/ext/` folder.
       2) Configuration and other files are placed in:
          - *domino_program_directory*`\idipwsync\` for Windows
          - *domino_data_directory*`/idipwsync/` for Linux- and UNIX-based platforms

    c. Configuring the Password Store:
       1) To configure the LDAP Password Store, see "Installing LDAP Password Store" on page 73.
       2) To configure the MQe Password Store, see "Installing MQ Everyplace Password Store" on page 86.

16. Delete the `idipwsync_install_r6.nsf` database. After the installation is complete, delete the `idipwsync_install_r6.nsf` database from the Domino server:
    a. In Lotus Domino Administrator, select **Files**.
    b. Right-click on the `idipwsync_install_r6` database and select **Delete Database**.
    c. Click **OK** in the **Confirm Database Delete** screen.

## Deployment on a Domino Domain with multiple Domino R6 Servers

In environments with multiple Domino Servers the Password Synchronizer is installed on each Domino Server which is a Primary Domino Directory Server in the Domino Domain.

The Password Synchronizer is not installed on Domino Servers which are Configuration Only Directory Servers.

The installation on the Primary Domino Directory Servers is performed as follows:

1. On the Primary Domino Directory Server that is the Administration Server for the Domino Directory, perform full installation of the password synchronizer as described in the Deployment on a single Domino R6 Server section of this document.
2. For all the other Primary Domino Directory Servers, do the following:
   a. Run the Password Synchronizer installer to install the necessary files.
   b. Force replication with the first Primary Domino Directory Server where a full setup is performed:
      1) In Lotus Domino Administrator select **Server**.
      2) Select **Status**.
      3) In the right-hand panel, select **Server/Replicate** .
      4) In the **Which server do you want to replicate with?** field, enter the name of the first Primary Domino Directory Server where a full setup is performed.
      5) Click **Replicate**.
      6) Click **Done**.
   c. The following setup instructions refer to the setup steps from the "Deployment on a single Domino R6 Server" on page 46 section previous:
      1) Skip steps 1, 2, 3 and 4.

         Domino Directory replication propagates the design updates from the first Primary Domino Directory Server where a full setup is performed.
      2) Skip steps 5 and 6.

         The **IDIPWSyncAdminRequestAgent** is triggered only on the Administration Server for the Domino Directory.
      3) Perform step 7, but skip step 7.a. (creation of a secret key).

         Use the secret key created when setting up the Password Synchronizer on the first Primary Domino Directory Server.
      4) Perform steps 8, 9, 10, 11, and 12.
      5) Skip step 13.
      6) Perform step 14, skipping step 14.a. (the creation of the **IDIPWSync** group).
      7) Perform step 15 and 16.

## Partial Deployment of the Password Synchronizer

The Domino Password Synchronizer intercepts both administrative password resets (when an administrator edits a user's person document) and normal password changes (when a user changes his own password through the **Change Password** Web form from `domcfg.nsf` or through iNotes).

Each one of these two features (intercepting administrative password resets and user password changes) can be installed and used independent of the other.

1. Install Domino Password Synchronizer that only intercepts administrative password resets (performed through the Lotus Domino Administrator or through the Web browser interface):

   To install a password synchronizer that will only intercept administrative password resets, perform all the steps from the "Deployment on a single Domino R6 Server" on page 46 section of this document, except steps 5 and 6.

   Steps 5 and 6 install the agent that intercepts normal user password changes.

   When installing the solution on a Domino Domain with multiple Domino Servers, follow the instructions from the "Deployment on a Domino Domain with multiple Domino R6 Servers" on page 57 section of this document, but do not perform steps 5 and 6 when installing the synchronizer on the Administration Server.
2. Install Domino Password Synchronizer that only intercepts normal user password changes (performed through the **Change Password** Web form from `domcfg.nsf` or through iNotes):

   To install a password synchronizer that intercepts normal user password changes only, perform the following steps from the "Deployment on a single Domino R6 Server" on page 46 section of this document: 1, 2, 5, 6, 9, 10, 11, 12, 15 and 16. The steps skipped (3, 4, 7, 8, 13 and 14) are not performed because they are only necessary for interception of administrative password resets.

   When installing the solution on a Domino Domain with multiple Domino Servers, it is only necessary to perform the previous subset of installation steps on the Primary Domino Directory Server which is the Administration Server for the Domino Directory. No installation on the other Domino Servers in the Domino Domain is necessary.

## Configuration: Synchronizing a unique user identifier

The Domino Password Synchronizer supplies an option to synchronize password changes using a unique User Identifier. The User Identifier, supplied by Domino, uniquely identifies users within their corresponding Domino Server.

**use.uniqueID**

This property enables users to be synchronized using a unique User Identifier obtained from Domino.

When this property is set to *true*, the Password Synchronizer performs synchronizations using the Domino supplied user identifier. A limitation of this feature is that users cannot be synchronized prior to being allocated a unique identifier by Domino. This restricts Password Synchronization to password changes that occur after an initial user account has been created.

When this property is set to *false*, the Password Synchronizer performs synchronizations using the user's username.

## Using the Password Synchronizer

### Only certain password change mechanisms are intercepted

When using the Domino HTTP Password Synchronizer, be aware that only certain password change mechanisms are intercepted. These are the mechanisms listed previous:

- Editing the Person document through the Lotus Domino Administrator

- Editing the Person document through the Web browser
- Using the Change Password Web form from `domcfg.nsf`
- Using iNotes

**Note:** Password changes performed through any other interfaces are not intercepted. For example, if passwords are changed through LDAP, or a Notes-Internet password synchronization is enabled, the Domino HTTP Password Synchronizer is not triggered and these password changes are not synchronized.

## Administrative password resets

The Domino HTTP Password Synchronizer is triggered when a user's Person document is edited and saved and the **Internet password** field of the Person document has been changed.

When synchronizing this type of password change (administrative password reset), the Domino HTTP Password Synchronizer hooks into the internal Domino logic before the password change is committed in Domino. If the Password Synchronizer successfully stores the changed password in the Password Store, the password change is performed in Domino. If the Password Synchronizer cannot store the changed password in the Password Store (for any reason), the password change is not performed in Domino and all other changes to the Person document are also rejected.

**Note:** This only applies for password changes performed through the Lotus Domino Administrator. After entering the new password value in the **Internet password** field, you must not switch from the **Basics** page of the opened Person document. If you switch to another page of the Person document before saving the changes, the password is hashed, the Domino HTTP Password Synchronizer is not able to get and no synchronization is triggered in this case.

## Normal user password changes

The Domino HTTP Password Synchronizer is triggered after a user changes his own password through the Password Change Web form or through iNotes. In both cases an administration request document (**Change HTTP password in Domino Directory**) is posted in the administration requests database. The Password Synchronizer is triggered after a document of this type is successfully processed by the Administration Process in Domino. At this stage the password change is already committed in Domino. If the Password Synchronizer successfully stores the password change in the Password Store, this administration request is marked as processed, so the administration request is not processed again the next time the Password Synchronizer is triggered. If the Password Synchronizer cannot store the password change in the Password Store (for any reason), the administration request is not marked as processed, so the Password Synchronizer attempts to process the administration request again the next time it is triggered.

**Note:** To enable the Change Password Web form, some setup is necessary in Domino (the Domino Configuration database DOMCFG.NSF must be created and session-based Web authentication must be enabled). For more information see the following articles in the Lotus Domino Administrator help: **Creating the Domino Configuration database** and **Setting up session-based name-and-password authentication**.

The component of the Domino HTTP Password Synchronizer that handles password change admin requests is a Domino agent named **IDIPWSyncAdminRequestAgent**. The **IDIPWSyncAdminRequestAgent** is a scheduled agent that is automatically (but not immediately) run after documents are created or changed in the administration requests database. It is the Agent Manager process that schedules what time after the actual document change that the agent is run. The Agent Manager checks two Domino Server parameters:

**AMgr_DocUpdateEventDelay**
: Specifies the delay time, in minutes, that the Agent Manager schedules a document update-triggered agent after a document update event. The default is 5 minutes. The delay time ensures the agent runs no more often than the specified interval, regardless of how frequently document update events occur.

**AMgr_DocUpdateAgentMinInterval**
: Specifies the minimum elapsed time, in minutes, between executions of the same document update-triggered agent. This lets you control the time interval between executions of a given agent. Default is 30 minutes.

The default values of these parameters mean that the agent is run 5 minutes after an admin request is created or changed, but no sooner than 30 minutes after a previous run of the same agent.

The **AMgr_DocUpdateEventDelay** and **AMgr_DocUpdateAgentMinInterval** parameters can be changed by editing the `NOTES.INI` file of the Domino Server (if the parameters are not specified there, you can add them, each on a separate line).

**Note:** These parameters affect all document update-triggered agents and setting low values can result in decreased server performance.

Admin requests stay in the administration requests database for a certain amount of time after they have been posted or last changed. Default value is 7 days (more than any rational values for the **AMgr_DocUpdateEventDelay** and **AMgr_DocUpdateAgentMinInterval** parameters). Do the following to check or change the garbage collection interval:

1. In Lotus Domino Administrator, select **Files**.
2. Right-click the **Administration Requests** database.
3. Select **Properties**.
4. Click **Replication Settings**.
5. Select **Space Savers**.

The value of interest is **Remove documents not modified in the last # days**.

When run, the agent processes in a batch all new password changes. It processes 5000 password changes at most. If more than 5000 password changes have been performed since the last run of the agent, it only processes 5000 password changes. The other password changes are processed during subsequent agent runs.

Another important Domino Server parameter that affects the behavior of the **IDIPWSyncAdminRequestAgent** is the **Max LotusScript/Java execution time**. This parameter has daytime and nighttime values that specify the maximum time an agent is enabled to run in the corresponding portion of the day. Defaults are 10 minutes for daytime and 15 minutes for nighttime. If the agent exceeds this time frame, it is stopped, and the unprocessed password changes are processed in subsequent runs. Change these values by editing the **Max LotusScript/Java**

**execution time** fields in the Server Document, section Server Tasks/Agent Manager. Note however that these settings affect all Java and LotusScript agents.

## Secure password transfer

Secure communication is achieved by enabling SSL for the Web-based mechanisms for password change (editing Person documents through the browser, using the Change Password Web form and using iNotes).

When editing Person documents through the Lotus Domino Administrator client, communication is secured by enabling port encryption in Domino.

For instructions on how to configure port encryption for Domino, see "Setup" on page 47 (specifically step 8).

## The Proxy Process

After the password is intercepted (in any of the supported password change mechanisms), it is always passed to the Proxy Process of the Domino HTTP Password Synchronizer. The Proxy Process instantiates a Password Store and uses it to store the password data. The Proxy Process is a Java Domino Server task. It is started by the Domino Server on startup and is stopped when the Domino Server stops. If necessary, the Proxy Process can be stopped and started manually from the Domino Server Console:

To manually stop the Proxy Process, enter the following Domino Command: **tell IDIPWSync quit**

To manually start the Proxy Process, enter the following Domino Command: **load runjava com.ibm.di.plugin.pwsync.domino.DominoProxy**

You can check whether the Proxy Process is started by entering the following command on the Domino Console: **show tasks**

If the Proxy Process is started, a line for the **IDI Password Sync** task (which is the Password Synchronizer Proxy Process) appears in the list. For example: **IDI Password Sync Listen for connect requests on TCP Port:19003**

## Synchronizing the access to the Password Store

Several password changes in Domino can be made at the same time from multiple users and from different interfaces. The Domino HTTP Password Synchronizer works with multiple threads of execution and attempts multithread access to the Password Store.

In cases when multithread access to the Password Store might be a problem (such as when MQe Password Store is used), you can synchronize the access to the Password Store. The configuration file of the Domino HTTP Password Synchronizer ididompwsync.props contains a property named **proxy.syncStoreAccess**. Set this property to **true** if you want to synchronize the access to the Password Store. Set this property to **false** if you want to enable multithread access to the Password Store.

**Note:** It is recommended to set **proxy.syncStoreAccess** to **true** when using the MQe Password Store because MQ Everyplace QueueManagers are not thread-safe. You can safely use multithread access with the LDAP Password Store.

## Changing the Password Store

The Password Store used by the Domino HTTP Password Synchronizer can be changed at any time after the initial deployment of the solution:

- Switching the Domino HTTP Password Synchronizer to use the LDAP Password Store:
  1. Make sure the LDAP Password Store is configured. The LDAP Password Store configuration file must be placed in the same folder where the Domino HTTP Password Synchronizer config file (**ididompwsync.props**) is placed.
  2. In the Domino HTTP Password Synchronizer config file (**ididompwsync.props**), place the following value for the **proxy.storeClassName** property:

     ```
     proxy.storeClassName=com.ibm.di.plugin.pwsync.LDAPPasswordSynchronizer
     ```

     The class **com.ibm.di.plugin.pwsync.LDAPPasswordSynchronizer** is included in the `proxy.jar` file shipped with the LDAP Password Store.
  3. Restart the Proxy Process of the Domino HTTP Password Synchronizer (see "The Proxy Process" on page 61).
- Switching the Domino HTTP Password Synchronizer to use the MQe Password Store:
  1. Make sure the MQe Password Store is configured. The MQe Password Store configuration file must be placed in the same folder where the Domino HTTP Password Synchronizer config file (**ididompwsync.props**) is placed.
  2. In the Domino HTTP Password Synchronizer config file (**ididompwsync.props**), place the following value for the **proxy.storeClassName** property:

     ```
     proxy.storeClassName=com.ibm.di.plugin.mqe.store.MQePasswordStore
     ```

     The class **com.ibm.di.plugin.mqe.store.MQePasswordStore** is included in the `mqepwstore.jar` file shipped with the MQe Password Store.
  3. Restart the Proxy Process of the Domino HTTP Password Synchronizer (see "The Proxy Process" on page 61).

# Chapter 7. Password Synchronizer for UNIX and Linux

This chapter describes the configuration and operation of the IBM Tivoli Directory Integrator PAM Password Synchronizer. This chapter contains the following sections:

## Overview

The Pluggable Authentication Modules (PAM) architecture on UNIX systems, provides an extendable design to enable customized behavior with respect to user authentication. The PAM Password Synchronizer Plug-in leverages the UNIX PAM architecture to enable password change notifications to be propagated to the Plug-in Password Store.

There are numerous online documentation sites describing all aspects of PAM. The AIX online manual has a good overview description at the following link:

```
http://publib16.boulder.ibm.com/pseries/en_US/aixbman/security/
    pam_overview.htm#plugauthmod.
```

## Supported Platforms

The PAM Password Synchronizer is available for the following platforms:

- Solaris 9 (32 bit and 64 bit)
- Solaris 10 (64 bit)
- AIX 5.2 (5200-08) (32 bit and 64 bit)
- AIX 5.3 (5300-03) (32 bit and 64 bit)
- HP-UX 11iv2 (11.23) (32 bit and 64 bit)
- RHEL AS 3.0 (IA32)
- RHEL AS 4.0 (IA32, AMD64/EMT64)
- Red Hat Desktop 4.0 (IA32) SLES 8/UL 1.0 (IA32)
- SLES 9 (IA32, AMD64/EMT64)
- Novell Desktop 9.0 (IA32)
- RedFlag Advanced Server 4.1 (IA32)

## Deployment

This section describes the deployment steps required for the PAM Password Synchronizer.

## Deploying the Password Synchronizer for UNIX and Linux

### Included files

The following files are included with the distribution package:

**libpamtivoli611.so**
: Contains the implementation which interfaces with IBM Directory Server account password change function for 32 bitLinux and Solaris UNIX environments.

**libpamtivoli611_64.so**
: Contains the implementation which interfaces with IBM Directory Server account password change function for 64 bitLinux and Solaris UNIX environments

**libpamtivoli611.a.so**
: Contains the implementation which interfaces with IBM Directory Server account password change function for 32 bitAIX UNIX environments.

**libpamtivoli611_64.a.so**
: Contains the implementation which interfaces with IBM Directory Server account password change function for 64 bitAIX UNIX environments.

**libpamtivoli611.sl**
: Contains the implementation which interfaces with IBM Directory Server account password change function for 32 bitHP-UX UNIX environments.

**libpamtivoli611_64.sl**
: Contains the implementation which interfaces with IBM Directory Server account password change function for 64 bitHP-UX UNIX environments.

**pampwsync.props**
: Configuration file required when using the LDAP Password Store.

**idicryptokeys.sh**
: Utility script file that creates test/demo keystore file.

**mqeconfig.props**
: Configuration file required when using the MQe Password Store.

**JAR Files**
: Contain the Password Synchronizer functionality that interfaces with the LDAP and MQe Password Stores: `proxy.jar`, `stopproxy.jar`, `mqepwstore.jar`, `idipwstore.jar`, `idipwcrypto.jar`, `mqeconfig.jar`, `auibase.jar`, `tdipwflt.jar`, `ibmjms.jar`, `MQeBase.jar`, `MQeJMS.jar`, `MQeSecurity.jar`, `MQeJMX.jar`

**LDIF Files**
: Contain the LDAP schema definition required for the specific Directory Server used as the LDAP Password Store: `ibm-diPersonSchemaForAD.ldif`, `ibm-diPersonForSunDS.ldif` , `ibm-diPerson_oc.ldif`, `ibm-diPerson_z.ldif`.

The installation creates a configuration file and places all the other required files in the appropriate directory structures. The .jar files are located in *`install_dir`*`/jvm/jre/lib/ext/`, where *install_dir* is the chosen installation directory. The .so/.a.so/.sl, .sh and .props files are located in *`install_dir`*`/PAM`.

The location binary ( `libpamtivoli611.so`, `libpamtivoli611.a.so` or `libpamtivoli611.sl`) will vary between platforms. The following table shows the location of the binary on various platforms, to which you should copy the respective PAM Password Synchronizer library:

| Operating System | PAM Binary Location |
|---|---|
| AIX 5.2 or greater | /usr/lib/security/libpamtivoli611.so |
| HP-UX 11.23 | /usr/lib/security/libpamtivoli611.sl |
| Solaris 8 | /usr/lib/security/libpamtivoli611.so |
| Solaris 9 | /usr/lib/security/sparcv9/libpamtivoli611.so |
| Linux | /lib/security/libpamtivoli611.so |

## Registering the Password Synchronizer for UNIX and Linux plug-in within PAM

To register the plug-in, edit the PAM configuration file. The table below shows the standard location of both PAM configuration files on various platforms. Your individual PAM configuration may cause the PAM password module configuration to be a different file. You should check with your system administrator if either these files do not exist, or if the added Password Synchronization module is not being invoked.

**Note:** Older versions of PAM on UNIX used the configuration file `/etc/pam.conf`. This file is now deprecated and all PAM configuration files should now be located in `/etc/pam.d` for modules that rely on PAM. The PAM configuration file for the password change module should be located in this directory.

| Operating System | PAM Configuration File |
|---|---|
| AIX 5.2 or greater | /etc/pam.conf or /etc/pam.d/system-auth |
| HP-UX 11.23 | /etc/pam.conf or /etc/pam.d/system-auth |
| Solaris 8 | /etc/pam.conf or /etc/pam.d/system-auth |
| Solaris 9 | /etc/pam.conf or /etc/pam.d/system-auth |
| Linux | /etc/pam.conf or /etc/pam.d/system-auth (RHEL 4)<br>/etc/pam.conf or /etc/pam.d/common-password (SLES 10) |

The primary component of external system configuration is the PAM configuration file. Since the purpose of the plug-in is to intercept password events, a line similar to the following should be added to the PAM configuration file. If the PAM module is being stacked with other PAM modules, then the Tivoli module should usually be the last in the stack. That way, the module can be sure that previous "required" modules have returned a success status before PAM called the Tivoli module.

**Note:** The above table list **system-auth** as the PAM configuration file in the `/etc/pam.d` directory. In actual fact, the configuration file `/etc/pam.d/passwd` is the main configuration file for password setting and changing. On most operating systems, the standard PAM install sets up `/etc/pam.d/passwd` to use `/etc/pam.d/system-auth` for defining the actual PAM modules use for password setting and changing. For example on RHEL 4, the delegation in the `/etc/pam.d/passwd` file might look as follows.

```
password  required  pam_stack.so  service=system-auth
```

If your PAM `/etc/pam.d/passwd` configuration file has delegated to **system-auth**, then you must add the configuration entry into `/etc/pam.d/system-auth`.

There are exceptions to the placement of the Tivoli module last in the stack:

- If there are modules above the Tivoli module in the stack, and they are marked as *sufficient*, then they must be changed to *required* to ensure that the Tivoli module is called. For example on RHEL 4 Linux you may find that the pam_unix module is marked as *sufficient* (standard installation). This means that if the result of the pam_unix module is successful, then no proceeding password modules will be invoked. To ensure that the Tivoli module is called, the pam_unix module must be changed to *required* and it must appear before the Tivoli module in the stack.
- If you have modules for error processing only, such as pam_deny, then they should follow the Tivoli module, and the Tivoli module should be then marked as *sufficient*.

Enter the following (as one line) into your system's PAM configuration file (see the table above):

**Using pam.conf**

```
passwd password required pam_binary_dir/libpamtiv.so use_first_pass
     install_directory/PAM/pampwsync.props
```

**Using the password module PAM configuration file in the `/etc/pam.d` directory (one of passwd, system-auth, common-password):**

```
passwd password required pam_binary_dir/libpamtivoli611.so use_first_pass
     install_dir/PAM/pampwsync.props
```

**Note:** For information on enabling PAM on AIX 5.2 or AIX 5.3 see Appendix A. AIX 5.2 PAM Configuration and Appendix B. AIX 5.3 PAM Configuration.

## Copy the PAM Module to the system

1. Place the 32-bit version of the PAM Module into the `/usr/lib/security` directory.
2. Place the 64-bit version of the PAM Module into the `/usr/lib/security/64` directory.
3. Ensure that the 32-bit and 64-bit Module share the same file name. Rename the 64-bit library by removing the "_64" part of the name. This ensures the one entry in the PAM .conf file will be relevant for both 32 and 64 bit modes. For example:

   ```
   /usr/lib/security/libpamtivoli.so
   /usr/lib/security/64/libpamtivoli.so
   ```

   **Note:** You should only use the file name of the modules (both should have the same filenames), not a fully qualified path to the module, for the **module_path** section of the Password Synchronizer PAM Module entry. When the PAM Framework loads your module, it will either use the 32-bit or 64-bit version, depending on the Operating System mode. If the Operating System is in 32-bit mode, then the path to the Password Synchronizer PAM module (`pamlibtivoli.so`) will be resolved to `/usr/lib/security/pamlibtivoli.so`. However, if the Operating System is in 64-bit mode, the path to the module will be resolved to `/usr/lib/security/64/pamlibtivoli.so`. Specifying an absolute path to the Password Synchronizer PAM Module binds the PAM Framework to one specific mode only (32-bit or 64-bit).

## File Permissions

In order to ensure the password synchronizer library can be executed by all users performing password changes, set the permissions of the 32-bit and 64-bit modules as follows:

```
chmod 555 /usr/lib/security/libpamtivoli.so
chmod 555 /usr/lib/security/64/libpamtivoli.so
```

It may also be necessary to set group ownership to ensure all users have authorization required to execute the Password Synchronizer. If required, set group ownership with a command such as the following for the 32-bit and 64-bit modules:

```
chown root:system /usr/lib/security/libpamtivoli.so
chown root:system /usr/lib/security/64/libpamtivoli.so
```

**Note:** A group other than `system` may be required, depending on your operating system configuration. In general the ownership and permissions should match that of the other modules in your `/lib/security/$ISA` directory.

## Configuration

This section describes the configuration steps required for the PAM Password Synchronizer.

### Configuring the Password Synchronizer for UNIX and Linux

The configuration file `/`*`install_directory`*`/PAM/pampwsync.props` contains properties in the format *`property_name=property_value`*, with each property placed on a single line.

Specify the following properties (property names are case sensitive):

**jvmPath**
: The path to the Java Virtual Machine executable (Java), used to run the Java layer (for example, *`install_dir`*`/jvm/jre/bin`).

**jvmClassPath**
: The Java classpath of the Java layer. Contains the path to the JAR files needed by the Password Synchronizer and the Password Store.

**syncClassName**
: The custom synchronization class that implements the **com.ibm.di.plugin.pwsync.IPasswordSynchronizer** interface and stores the passwords. Valid values are:

- com.ibm.di.plugin.pwsync.LDAPPasswordSynchronizer
- com.ibm.di.plugin.mqe.store.MQePasswordStore
- com.ibm.di.plugin.pwsync.LDAPPasswordSynchronizerITIMDecorator
- com.ibm.di.plugin.mqe.store.MQePasswordStoreITIMDecorator

The last two are the decorator implementations which interface with ITIM Password Strength Servlet before delegating to a contained instance of the first two.

**serverPort**
: The port on which the C and Java layers communicate.

**logFile**
: The C layer log file, used to log synchronization failures.

**syncBase**
This configuration parameter is currently not relevant for the PAM Password Synchronizer. You should clear this value if it is set, although having a value will have no affect on the operation of the PAM Password Synchronizer.

**javaLogFile**
The log file of the Java layer.

**checkRepository**
Specify true or false to correspondingly turn on or off the check for repository availability. When this option is turned on and the repository is not available, the Password Synchronizer cancels the password update in PAM.

The following is a (UNIX) sample of the PAM Password Synchronizer configuration file, `pampwsync.props`:

```
jvmPath=install_dir/jvm/jre/bin
jvmClassPath=install_dir/PAM;install_dir/PAM/jars/
   proxy.jar;install_dir/PAM/jars/ibmjms.jar;install_dir/
   PAM/jars/idipwcrypto.jar;install_dir/PAM/jars/idipwstore.jar;
   install_dir/PAM/jars/auibase.jar;install_dir/
   PAM/jars/MQeBase.jar;install_dir/PAM/jars/mqeconfig.jar;
   install_dir/PAM/jars/MQeJMS.jar;install_dir/PAM/
   jars/mqepwstore.jar;install_dir/PAM/jars/MQeSecurity.jar;
   install_dir/PAM/jars/stopproxy.jar;install_dir/PAM
   /jars/tdipwflt.jar;install_dir/IDS/jars/MQeJMX.jar
syncClassName=com.ibm.di.plugin.pwsync.LDAPPasswordSynchronizer
serverPort=19014
logFile=install_dir/PAM/pam_pw_sync.log
syncBase=
javaLogFile=install_dir/PAM/pam_pws_java.log
checkRepository=true
```

**Note:** *install_dir* is a variable. Do not copy and paste this example without first replacing variables with accurate data.

## Enabling PAM Debug

The PAM library can provide debugging information during execution. After you enable the system to collect debug output, you can use the information gathered to track PAM invocations. You can also use it to determine failure points in the current PAM setup. To enable PAM debug output, follow these steps:

1. Update your PAM configuration entry in the appropriate PAM configuration file to include the debug parameter. In the `system-auth` file entry, for example, the entry would appear as follows:
   ```
   password required pam_binary_dir/libpamtiv.so debug use_first_pass
      install_dir/PAM/pampwsync.props
   ```
2. Create an empty file named `/etc/pam_debug`. The PAM library checks for the existence of `/etc/pam_debug` file. If found, it enables syslog output.
   ```
   # touch /etc/pam_debug
   ```
3. Edit the `/etc/syslog.conf` file to contain the appropriate entries for the desired levels of messages. To capture debug information for authentication calls, use **auth.debug** and send the output to a file. The following example line is in the `syslog.conf` file that sends the authentication debug information to a file:
   ```
   auth.debug /tmp/syslog_auth.log
   ```
4. Restart the syslogd daemon so that the configuration changes made in `/etc/syslogd.conf` are recognized:

```
# stopsrc -s syslogd # startsrc -s syslogd
```

5. When an authentication action occurs, including PAM authentications, debug messages are collected in the output file defined in the `/etc/syslog.conf` configuration file, in this case `/tmp/syslog_auth.log`. In some cases, the PAM module logs may be found in a file such as `/var/logs/secure`.

## Integrating the Password Store

The following sections describe how to integrate the Password Store.

### Integrating with the LDAP Password Store

To integrate the PAM Password Synchronizer with the LDAP Password Store, do the following:

1. In the PAM Password Synchronizer config file (pampwsync.props), place the following value for the **syncClassName** property:

   ```
   syncClassName=com.ibm.di.plugin.pwsync.LDAPPasswordSynchronizer
   ```

   The class **com.ibm.di.plugin.pwsync.LDAPPasswordSynchronizer** is included in the `proxy.jar` file, which is shipped with the LDAP Password Store.
2. Follow the configuration instructions of the LDAP Password Store. See LDAP Password Store.
3. Place the LDAP Password Store configuration file in the folder where you installed the IBM Directory Server Password Synchronizer.

For instructions on how to configure the properties file for the LDAP Password Store, see Preparing the LDAP Password Store properties file (idipwsync.props).

### Integrating with the MQe Password Store

To integrate the PAM Password Synchronizer with the MQe Password Store, do the following:

1. In the PAM Password Synchronizer config file (pampwsync.props), place the following value for the **syncClassName** property:

   ```
   syncClassName=com.ibm.di.plugin.mqe.store.MQePasswordStore
   ```

   The class **com.ibm.di.plugin.mqe.store.MQePasswordStore** is included in the `mqepwstore.jar` file shipped with the MQe Password Store.
2. Follow the configuration instructions of the MQe Password Store (see MQ Everyplace Password Store).
3. Place the configuration file of the MQe Password Store in the folder where you installed the PAM Password Synchronizer (*install_directory*/PAM).

**Note:** You can change the Password Store used by the PAM Password Synchronizer. If the corresponding Password Store is installed and configured, do the following:

1. Comment out the PAM Password Synchronizer configuration entry within the system's PAM configuration file; this will stop passwords being synchronized until the Password Store is reconfigured.
2. Stop the Java Proxy (the Java Proxy will restart on the first Password Synchronization).
3. Edit the PAM Password Synchronizer configuration file and set the **syncClassName** accordingly.
4. Un-comment the PAM Password Synchronizer configuration entry within the system's PAM configuration file.

### Stopping the Java layer (Java Proxy)

The Java Layer of the PAM Password Synchronizer is not automatically terminated once the PAM Password Synchronizer is disabled (via removing its configuration entry from the system PAM configuration file). To terminate the Java Layer use the **StopProxy** utility. It is a standalone Java application that sends a command to the PAM Password Synchronizer Java layer requesting termination.

The **StopProxy** is distributed in the `proxy.jar` `java` archive which is placed in *`install_directory`*`/PAM/jars`. Do the following to use the **StopProxy** (as one line):

```
"install_directory/jvm/jre/bin/java"
     -DpropertiesFile=password_synchronizer_config_file
     -cp "install_directory/PAM/jars/proxy.jar"
     com.ibm.di.plugin.pwsync.StopProxy
     port [hostIP]
```

where:

- *password_synchronizer_config_file* is the fully qualified file system path of the password synchronizer configuration file (`pampwsync.props`).
- *port* is a required parameter and represents the port used for communication between the C and Java layer (use the value of the **serverPort** configuration parameter).
- *[hostIP]* is an optional parameter that, when specified, represents the IP address of the machine where the PAM Password Synchronizer is deployed (when omitted, a default value of *localhost* is assumed).

## Using the Password Synchronizer

One of the configuration properties of the IBM Tivoli Directory Server Password Synchronizer is of particular interest and directly affects the password synchronization logic:

**checkRepository**

This property enables turning on or off the functionality that checks for availability of the Password Storage.

When this property is set to *true*, the Password Synchronizer first checks whether the Password Storage is available. If it is available, the password is changed in the directory, then the password is sent to the Password Storage. If the check indicates that the storage is not available, the LDAP operation (a part of which is the password update) is rejected on the IBM Tivoli Directory Server.

When the **checkRepository** property is set to *false*, the Password Synchronizer performs no checks for storage availability. The password update is performed in the directory first, then an attempt is made to store it in the Password Storage. If the password cannot be stored, a message is logged in the log file (pointed to by the logFile property) to indicate that password synchronization for this user failed.

**Note:** The check for availability of the Password Storage works with all Password Store components.

## Stopping the Proxy Layer

The PAM Password Synchronizer consists of two layers: an PAM plug-in which is hooked into the host Operating System's PAM interface, and a Java Proxy Layer. The plug-in intercepts password updates and sends them to the Proxy Layer. The Proxy Layer instantiates the Password Store component on startup and transmits all password updates received by the plug-in to the Password Store.

The Proxy Layer is started automatically by the PAM plug-in when the PAM plug-in receives a password synchronization request (only when the Java Layer is not already running). However, it is not stopped when the PAM is disabled. The Proxy Layer must be stopped explicitly when the PAM Password Synchronizer plug-in is disabled.

If you do not stop the Java Layer explicitly, the PAM Password Synchronizer does not start properly the next time the PAM plug-in is activated.

Use the **StopProxy** utility included in the PAM Password Synchronizer to stop the Java Layer. It is a standalone Java application and its usage is described in "Stopping the Java layer (Java Proxy)" on page 70.

## Reference Material

For information on enabling PAM on AIX 5.2 or AIX 5.3 see Appendix A. AIX 5.2 PAM Configuration and Appendix B. AIX 5.3 PAM Configuration.

The following publications have been found useful in configuring PAM on AIX:

- *AIX 5L™ Version 5.2 Security Guide* - Chapter 7. *Pluggable Authentication Module* contains a guide to configuring PAM on AIX 5.2.
- *AIX 5L Version 5.2 Security Supplement* - Chapter 4. *Pluggable Authentication Module* contains instructions for configuring PAM on AIX 5.2 to use a third-party LDAP Authentication Module.
- *AIX 5L Differences Guide Version 5.2 Edition* - Describes the support differences between AIX 5.1 and AIX 5.2 in regard to PAM support.

# Chapter 8. LDAP Password Store

The LDAP Password Store provides the function necessary to store the intercepted user passwords in an LDAP directory server.

This chapter contains the following sections:
- "Supported Directories"
- "Installing LDAP Password Store"
- "Using the Password Store" on page 82

## Supported Directories

The LDAP Password Store is available on the following directories:
- IBM Tivoli Directory Server
- Microsoft Active Directory
- Sun ONE Directory Server

## Installing LDAP Password Store

This section describes the LDAP Password Store installation process, including prerequisites and included files.

### Overview

IBM Directory Integrator LDAP Password Store provides the function necessary to store the intercepted user passwords in an LDAP directory server (repository or datasource). Supported directories include IBM Directory Server, Microsoft Active Directory and Sun ONE Directory Server.

The LDAP Password Store component of this package was created to support a growing number of IBM Directory Integrator plug-ins which intercept password changes for various products or platforms.

The following password synchronization plug-ins are available to intercept a user's password change request:

**IBM Directory Integrator Password Synchronizer for Windows**
Intercepts the Windows login password change.

**IBM Directory Server Password Synchronizer for Windows, UNIX and Linux**
Intercepts an IBM Directory Server password change.

**Sun ONE Directory Server Password Synchronizer for Windows, UNIX and Linux**
Intercepts the Sun ONE Directory Server password change.

**Domino Password Synchronizer for Windows, UNIX and Linux**
Intercepts changes of the HTTP password for Lotus Notes users.

**IBM Directory Integrator Password Synchronizer for UNIX and Linux**
Intercepts changes of UNIX and Linux user passwords.

These plug-ins all utilize the LDAP Password Store function which facilitates the secure propagation of the change to another LDAP server where it can later be manipulated by an IBM Directory Integrator AssemblyLine.

The ability to tailor the LDAP Password Store is accomplished using properties files which enable the specification of keystore files, certificates and credentials for SSL connections and the asymmetric encryption of password data. The property files also accommodate control of trace logging, and limited control of attributes used for storing captured passwords.

## Prerequisites

The LDAP Password Store requires JRE 1.5 (included).

## Included files

**idipwcrypto.jar**
Contains the asymmetric encryption support for LDAP Password Store.

**idipwstore.jar**
Contains the LDAP Password Store function.

**ibmjcefw.jar (Only installed with Domino Password Synchronizer)**
Used by the asymmetric encryption support.

**ibmjceprovider.jar (Only installed with Domino Password Synchronizer)**
Used by the asymmetric encryption support.

**ibmpkcs.jar (Only installed with Domino Password Synchronizer)**
Used by the asymmetric encryption support.

**ibmpkcs11.jar (Only installed with Domino Password Synchronizer)**
Used by the asymmetric encryption support.

**local_policy.jar (Only installed with Domino Password Synchronizer)**
Used by the asymmetric encryption support.

**US_export_policy.jar (Only installed with Domino Password Synchronizer)**
Used by the asymmetric encryption support.

**ibmjsse.jar (Only installed with Domino Password Synchronizer)**
Used by the SSL support.

**ibm-diPerson_oc.ldif**
Contains schema definition of ibm-diPerson object class for IBM Directory Server and most LDAP servers.

**ibm-diPersonForAD.ldif**
Contains schema definition of ibm-diPerson object class for Active Directory.

**ibm-diPersonForSunDS.ldif**
Contains schema definition of ibm-diPerson object class for Sun ONE Directory Server.

**ibm-diPerson_z.ldif**
Contains schema definition of ibm-diPerson object class for zLDAP Directory Server.

**idicryptokeys.bat (Windows), idicryptokeys.sh (UNIX and Linux)**
Contains example keytool command to generate keystore for password encryption.

## Installing LDAP Password Store

Do the following to set up and install the LDAP Password Store:

## Set up the LDAP server

The following instructions describe how to set up a sample environment using IBM Directory Server. This involves identifying a container where the object class containing the user ID and password is found or created.

Do the following to set up a sample environment using IBM Directory Server:

1. Define the suffix.
   a. Start **Directory Configuration**. Select **Start-->Programs-->IBM Directory Server x.x-->Directory Configuration**.
   b. Select **Manage suffixes** from the pane in the left.
   c. In the **Suffix DN** field add the suffix under which you store the password information (for example, `o=ibm,c=us`).
   d. Click **Add**.
   e. The new suffix is shown in the **Current suffix DNs** list. Click **OK**.
   f. Close the **Directory Configuration** tool.
2. Add the suffix data.
   a. Restart the IBM Directory Server.
   b. Using **IBM Directory Server Web Administration Tool**, select **Directory management-->Manage entries**.
   c. Click **Add**.
   d. Select **organization** from the structural object class list.
   e. Click **Next**.
   f. In the **Select auxiliary object classes** screen, click **Next**.
   g. In the **Enter the attributes** screen, clear the value of the Parent DN field.
   h. Enter the suffix name into the **Relative DN** field (for example, **o=ibm,c=us**).
   i. Enter the organization name into the **o** field (**ibm** in the previous example).
   j. Click **Finish**.
3. Add the domain object.
   a. Still using **IBM Directory Server Web Administration Tool**, select **Directory management-->Manage entries**.
   b. Select the suffix previously created in the previous step (**o=ibm,c=us**) by selecting the corresponding radio-button.
   c. Click **Add...**
   d. Select **domain** from the **structural object class** list.
   e. Click **Next**.
   f. In the **Select auxiliary object classes** screen, click **Next**.
   g. Enter the domain name in the **Relative DN** field (for example, **dc=mydomain**).
   h. Enter the domain name in the **dc** field (**mydomain** in the previous example).
   i. Click **Finish**.

   **Note:** The domain and suffix entered must also be included in the idipwsync.props file along with the other information (see "Preparing the LDAP Password Store properties file (idipwsync.props)" on page 78 for more details).
4. Define the **ibm-diPerson** object. From a machine with IBM Directory Server Client, issue the following command from the *install_directory* (as one line):

```
ldapmodify -c -h LDAP Hostname -D admin DN -w admin PW
       -f ibm-diPerson_oc.ldif
```

**Note:** You might see the following messages:

```
attribute type '1.3.18.0.2.4.155' already exists, add operation
failed.
```

or

`attribute type '0.9.2342.19200300.100.1.1' already exists, add operation failed.` You can ignore these messages, they indicate that these **secretKey** and **uid** attributes are already defined in your schema.

## Modifying the schema of zLDAP

**Note:** When configuring the LDAP server on z/OS, the LDAP server must be configured with a TDBM back end (this enables loading of the required LDIF file). Detailed instructions for setup and configuration of the IBM LDAP server on z/OS with a TDBM back end are beyond the scope of this guide. For further information on this issue, please see the document *z/OS Integrated Security Services LDAP Server Administration and Use* in the IBM z/OS online product library.

Modify the schema of zLDAP as follows

1. Definition of a suffix involves generating a new LDAP config and server JCL Jobs. This must be performed jointly by the LDAP administrator and the SYS Programmers.
2. While the suffix data does not need to be added, the base schema does need to be added to the defined suffix. The two base schema LDIF files in the `/usr/lpp/ldap/etc` directory, `schema.IBM.ldif` and `schema.user.ldif`, must be customized with the suffix from step 1 above, and then loaded.
3. If required, a domain can be defined by creating and loading a LDIF file that defines the domain.
4. Replace the contents of the `ibm-diPerson_z.ldif` file with the following:

```
# Module Name: IBM Tivoli Directory Integrator Password Synchronizer 6.1.1
# Dependencies:
#
dn:cn=schema,
changetype: modify
add: attributetypes
attributetypes: (
  1.3.18.0.2.4.2990
  NAME 'ibm-diExtendedData'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  DESC 'Extended data entry for synchronizing passwords with
    IBM Directory Integrator.'
  EQUALITY caseIgnoreMatch
  ORDERING caseIgnoreOrderingMatch
  SUBSTR caseIgnoreSubstringsMatch
  USAGE userApplications
  )
ibmattributetypes: (
  1.3.18.0.2.4.2990
  DBNAME( 'diExtendedData'  'diExtendedData' )
  ACCESS-CLASS normal
  )

dn:cn=schema,
changetype: modify
add: attributetypes
attributetypes: (
  1.3.18.0.2.4.3009
  NAME 'ibm-diPassword'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
  DESC 'Encrypted value of password to be synchronized.'
  EQUALITY octetStringMatch
  USAGE userApplications
  )
ibmattributetypes: (
  1.3.18.0.2.4.3009
  DBNAME( 'diPassword'  'diPassword' )
  ACCESS-CLASS normal
  )

dn:cn=schema,
changetype: modify
add: attributetypes
attributetypes: (
  1.3.18.0.2.4.3011
  NAME 'ibm-diUserid'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  DESC 'Userid  for synchronizing passwords with IBM Directory
    Integrator.'
  EQUALITY caseIgnoreMatch
  ORDERING caseIgnoreOrderingMatch
  SUBSTR caseIgnoreSubstringsMatch
  USAGE userApplications
  )
ibmattributetypes: (
  1.3.18.0.2.4.3011
  DBNAME( 'diUserid'  'diUserid' )
  ACCESS-CLASS normal
  )

dn:cn=schema,
changetype: modify
add: objectclasses
objectclasses: (
  1.3.18.0.2.6.547
  NAME 'ibm-diPerson'
  DESC 'Person entry for synchronizing passwords with IBM Directory
    Integrator.'
  SUP top
  STRUCTURAL
  MAY ( ibm-diExtendedData $ secretKey $ ibm-diPassword $ ibm-diUserid
     $ uid )
  )
```

5. Before the `ibm-diPerson_z.ldif` file can be loaded into the LDAP server, it must be customized to include the suffix created in step 1. This involves adding the suffix to the end DN's. For example, if your suffix was ″o=ibm,c=us″ then the DN lines would change from this:

   ```
   dn:cn=schema
   ```

   To this:

   ```
   dn:cn=schema,o=ibm,c=us
   ```

## Modifying the schema of Sun ONE Directory Server and Active Directory

1. Modify the LDAP schema of Sun ONE Directory Server. Issue the following command (as one line):

```
ldapmodify -c -h <LDAP Hostname> -D <admin DN> -w <admin PW>
       -f ibm-diPersonForSunDS.ldif
```

2. Modify the LDAP schema of Active Directory.
   a. Enable Active Directory schema modification by editing the Windows registry key:
      ```
      HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\NTDS\Parameters
      ```
      Add a **REG_DWORD** value named **Schema Update Allowed** with a value of **1** (or any value greater than **0**).
   b. Issue the following command to update the LDAP schema:
      ```
      ldifde -i -f ibm-diPersonSchemaForAD.ldif
      ```

## Preparing the LDAP Password Store properties file (idipwsync.props)

This section describes how to prepare the `idipwsync.props` file.

**Note:** The properties file must be named **idipwsync.props**. In order to be located by the program on startup, it must be placed in a directory that is in the CLASSPATH of the Password Synchronizer (the classpath of the IBM Directory Server Password Synchronizer is specified through the jvmClassPath property in the IBM Directory Server Password Synchronizer configuration file).

Execute the following command (one line) to generate a partial properties file with some basic settings:

```
install_directory\jvm\jre\bin\java
   com.ibm.di.plugin.idipwsync.GenPropertiesFile filepath
   ldapLoginPassword sslKeyStorePassword
   encryptKeyStoreFilePassword encryptKeyPassword
```

where

- *install_directory* is the directory where the plug-in is installed, for example: `C:\IBM\DiPlugins\IDS` or `C:\IBM\DiPlugins\IDI` (not `C:\IBM\DiPlugins`).
- *filepath* is the fully qualified filename (for example, *install_directory*\ idipwsync.props).
- *ldapLoginPassword* is the un-encoded password for directory access (for example, **secret**).
- *sslKeyStoreFilePassword* is the un-encoded password for the keystore file used for ssl connection (for example, **secret**).
- *encryptKeyStoreFilePassword* is the un-encoded password for the keystore file used for encrypting passwords (for example, **secret**).
- *encryptKeyPassword* is the un-encoded password for accessing the private key used for encrypting passwords (for example, **secret**).

For example,

```
install_directory\jvm\jre\bin\java com.ibm.di.plugin.idipwsync.GenPropertiesFile
      install_directory\idipwsync.props secret secret secret secret
```

yields the following file contents:

```
#IBM Directory Integrator LDAP Password Store Settings with
    Encoded Passwords
#Wed Jan 15 11:22:06 EST 2003
suffix=
logFilePath=
encryptKeyStoreFilePassword=0c0bf0e3146b
```

```
port=
encryptKeyStoreFilePath=
sslKeyStoreFilePath=
ssl=true
host=
sslKeyStoreFilePassword=0c0bf0e3146b
encrypt=true
InformationTrace=true
ldapLogInUserId=
DiagnosticTrace=true
waitForStore=true
WarningTrace=true
ldapLogInPassword=0c0bf0e3146b
encryptKeyStoreCertificate=
encryptKeyPassword=0c0bf0e3146b
ErrorTrace=true
```

**Note:** This utility does not set all the required properties. You must edit the information specific to your installation. Also, the *sslKeyStoreFilePath* is the path to a JKS file which you need to obtain or create using other utilities not included in this package. A keystore file is necessary only if you are enabling SSL on your Directory Server connection. This implementation assumes that a single JKS file contains the certificate for both the client and server if Server and Client Authentication is selected on the Directory Server's SSL Settings. The `ssl` property must be set to **true** to enable these properties.

If you choose to have the captured passwords stored in encrypted format, specify a JKS file for the value of *encryptKeyStoreFilePath*. The certificate alias containing the private-public key pair is specified as the value for *encryptKeyStoreCertificate*. The `encrypt` property must be set to **true** to enable these properties. See "About encrypting passwords" on page 81 for more information.

If you choose to code the properties file, or wish to later change either the *ldapLogInPassword*, *sslKeystoreFilePassword*, *encryptKeyStoreFilePassword*, or *encryptKeyPassword* you can use a second utility which encodes your new password, so you can copy it into the properties file.

To obtain an encoded version of your password, issue the following command (as one line):

```
install_directory\jvm\jre\bin\java com.ibm.di.plugin.idipwsync.EncodePW
      password
```

where *password* is your ASCII password (for example, **secret** ).

For example:

```
install_directory\jvm\jre\bin\java com.ibm.di.plugin.idipwsync.EncodePW secret
```

returns

```
0c0bf0e3146b
```

An example of a completed properties file for an SSL connection and password encryption looks like the following:

```
#IBM Directory Integrator LDAP Password Store Settings with
     Encoded Passwords
#Tue Jul 30 08:21:20 EDT 2002
host=gbdthst1
port=636
DiagnosticTrace=true
```

```
ErrorTrace=true
InformationTrace=false
WarningTrace=true
waitForStore=true
ldapLogInUserId=cn=root
ldapLogInPassword=0c0bf0e3146b
ssl=true
sslKeyStoreFilePath=c:\sync\keys.jks
sslKeyStoreFilePassword=0c0bf0e3146b
suffix=dc=carnd11,o=ibm,c=us
encrypt=true
encryptKeyStoreFilePath=c:\sync\cryptokeys.jks
encryptKeyStoreFilePassword=0c0bf0e3146b
encryptKeyStoreCertificate=cryptoCertName
encryptKeyPassword=0c0bf0e3146b
logFilePath=c:/sync/idipwsync.log
```

**Notes:**

1. To disable SSL, select a non-SSL port (for example, 389) and set **ssl=false**. When **ssl=false**, any value in *sslKeyStoreFilePath* and *sslKeyStoreFilePassword* is ignored.
2. To disable asymmetric password encryption, set **encrypt=false**. When **encrypt=false**, any value in *encryptKeyStoreFilePath* , *encryptKeyStoreFilePassword* , *encryptKeyStoreCertificate* and *encryptKeyPassword* is ignored.
3. The suffix keyword is used to identify the container where objects containing the user ID and new password value are found.
4. There are some additional optional keywords that can be used to override the default object class and attribute definitions provided. The following are the names of the properties that can be added in idipwsync.props and their associated default values:

   **schemaPersonObjectName**
   ibm-diPerson

   **schemaUseridAttributeName**
   ibm-diUserId

   **schemaPasswordAttributeName**
   ibm-diPassword

   **schemaExtendedDataAttributeName**
   ibm-diExtendedData

   Caution must be used to ensure that the names specified using these keywords match the schema you have defined for your directory server.
5. If multi-value password support is required, use the **ibm-diExtendedData** attribute for the **schemaPasswordAttributeName**. The properties file entry looks like the following:

   ```
   schemaPasswordAttributeName=ibm-diExtendedData
   ```
6. Another optional attribute, `delayMillis`, is used when the `waitForStore` property is set to **false**. When **waitForStore=false**, `delayMillis` specifies the number of milliseconds of delay prior to performing the store. If the IBM Directory Integrator Password Synchronizer for Windows is configured to use the LDAP Password Store and the LDAP Password Store is configured to store into Active Directory on the same machine where the Password Synchronizer is installed a deadlock can occur. To avoid the deadlock use this asynchronous mode of operation. In the asynchronous mode (`waitForStore=false`), the password catcher code which communicates with the Windows system returns control to Windows. After a short delay, the password store code which is running a separate thread attempts the store of the password update into

Active Directory. If **waitForStore=false** and no value is specified for `delayMillis`, then a default of **delayMillis=2000** is used. In this configuration, any password store failures are reported using the log file specified in the `logFilePath` property.

## About encrypting passwords

When the property `encrypt` is set to **true**, the *encryptKeyStoreFilePath*, *encryptKeyStoreFilePassword*, *encryptKeyStoreCertifcate* and *encryptKeyPassword* values must also be set. The password encryption and decryption functions provided in the `idipwcrypto.jar` file use RSA key algorithm. You can use the provided `idicryptokeys.bat` file to help build a test keystore for testing. If you use the `idicryptokeys.bat` file to create a test keystore, your values in the properties file look like the following:

**encryptKeyStoreFilePath=***anypath***/idicryptotest.jks**
where *anypath* is the location of the JKS file created by the `idicryptokeys.bat` file (see `-keystore` parameter setting in `idicryptokeys.bat`)

**encryptKeyStoreFilePassword=secret**
(see `-storepass` parameter setting in `idicryptokeys.bat`)

**encryptKeyStoreCertifcate=idicryptotest**
(see `-alias` parameter setting in `idicryptokeys.bat`)

**encryptKeyPassword=secret**
(see `-keypass` parameter setting in `idicryptokeys.bat`)

Information about keystores and keytools is available from the following sites:
- `http://www-128.ibm.com/developerworks/websphere/techjournal/0502_benantar/0502_benantar.html#sec2`
- `http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html`

The `java.security` file located in the *`install_directory`*`/jvm/jre/lib/security` directory has been set up to contain a reference to security provider `com.ibm.crypto.provider.IBMJCE` . The following is an example of how the relevant portion of the file might look:

```
                         :
                         :
                         :

# List of providers and their preference orders :
#
security.provider.1=sun.security.provider.Sun
security.provider.2=com.ibm.crypto.provider.IBMJCE

                         :
                         :
                         :
```

An example AssemblyLine which demonstrates the decryption of captured passwords is included in the IBM Directory Integrator installation. The AssemblyLine and a readme file are located in the *`install_directory`*`/examples/pwsync_decryption/` directory where *install_directory* is the install directory of the IBM Directory Integrator .

## Using the Password Store

For each user whose password has been intercepted, the LDAP Password Store maintains an LDAP entry in the storage LDAP directory (the container where the storage entries are added and modified is specified by the suffix property of the LDAP Password Store).

The entry kept in the storage directory always contains the passwords currently used by the original user on the Target System. To achieve this, the LDAP Password Store updates the state of the entry in the storage directory whenever the LDAP Password Store receives notification for password update from the Password Synchronizer.

The LDAP Password Store receives the following data from the Password Synchronizer:

- User identifier (a string)
- Type of the password modification
- A list of password values

**User Identifier**

The user identifier is used for the relative distinguished name of the entry stored in the LDAP directory. For example, if the user identifier is ″john″ and the suffix property value is ″dc=somedc,o=ibm,c=us″, then the distinguished name of the entry stored is ″ibm-diUserId=john, dc=somedc,o=ibm,c=us″.

Special attention is necessary when the LDAP Password Store is used with the IBM Tivoli Directory Server Password Synchronizer or with the Sun ONE Directory Server Password Synchronizer.

The Password Synchronizer reports the LDAP distinguished name of the user for which the password has been changed. For example, ″cn=john,o=somecompany,c=us″. The LDAP Password Store takes the first element of the distinguished name (″john″) to construct the distinguished name of the entry on the storage LDAP directory, for example, ″ibm-diUserId=john, dc=somedc,o=ibm,c=us″. Therefore the context information (department, company, country, and so forth) is lost. If there are two individuals on the Target System with equal names but in different departments, for example, ″cn=Kyle Nguyen,ou=dept_1,o=ibm,c=us″ and ″cn=Kyle Nguyen,ou=dept_2,o=ibm,c=us″, they are indistinguishable for the Password Store, and the Password Store acts as if they represent the same person.

**Type of password modification and List of password values**

The type of password modification indicates whether the password values have been replaced, or new values have been added, or certain values have been deleted. Using this information and the list of passwords representing the change, the Password Store duplicates the change on the entry in the storage directory.

The type of password modification makes sense only when the password can have multiple values (IBM Tivoli Directory Server, Sun ONE Directory Server). When the passwords on the Target System are single-valued (Windows), the password modification type is always **replace**.

When the password (with all its values) is deleted from the Target System, the entry in the storage directory is modified so that it does not have value for the LDAP attribute used to store the passwords.

## Possible password retrieval from IBM Tivoli Directory Integrator

Here is a possible mechanism for retrieving passwords stored in an LDAP Server by the LDAP Password Store:

An EventHandler is configured to listen for changes in the LDAP Directory used for storage. Whenever the EventHandler detects that an entry has been added or modified in the Password Store container, it starts an AssemblyLine, passing it identification of the modified entry. The AssemblyLine uses an LDAP Connector to read the modified entry, then decrypts the updated password values and propagates the values to systems that must be kept synchronized.

# Chapter 9. MQ Everyplace Password Store

This chapter contains the following sections:

## Overview

MQ Everyplace Password Store (MQe Password Store) provides the function necessary to store user passwords into IBM WebSphere MQ Everyplace and transfer user passwords from MQ Everyplace to IBM Tivoli Directory Integrator .

The MQe Password Store package consists of the Storage Component and the MQe Password Store Connector. The Storage Component is actually the Password Store invoked by the Password Synchronizer. The MQe Password Store Connector is a specialized Connector on the IBM Tivoli Directory Integrator side that can retrieve passwords stored into MQ Everyplace.

### Solution structure and workflow

Two MQ Everyplace QueueManagers are instantiated and configured: one on the Target System, and one on the IBM Tivoli Directory Integrator machine.

On the QueueManager on the IBM Tivoli Directory Integrator , a local queue is defined. On the QueueManager on the Target System, an asynchronous remote queue that references the local queue on the IBM Tivoli Directory Integrator QueueManager is defined. A connection and listener objects are defined in the QueueMangers to enable network communication.

The following is the workflow for the MQe Password Store:

1. The Password Synchronizer intercepts a password change and sends it to the Storage Component.
2. The Storage Component wraps the password into an MQe message and sends the message to the remote queue on the local QueueManager.
3. The MQe QueueManager on the Storage Component automatically sends the message to the QueueManager on the IBM Tivoli Directory Integrator .
4. The MQe Password Store Connector connects to the local QueueManager and reads the password update messages from the local queue.

### Supported WebSphere MQ Everyplace version

The MQe Password Store contains WebSphere MQ Everyplace v.2.0.2.5 embedded. No separate installation of WebSphere MQ Everyplace is necessary.

Part of the MQe Password Store deployment and configuration is the instantiation and configuration of the MQ Everyplace QueueManagers.

Once the MQe QueueManagers are instantiated and configured it is not recommended to change their configuration. If a change is necessary, the preferred method is to delete the QueueManager and recreate it again following the MQe Password Store deployment instructions. If however, for any reason, you are going to use an MQe administration tool to change QueueManagers settings, make sure this tool is compatible with QueueManagers created with MQ Everyplace v.2.0.2.5.

## Installing MQ Everyplace Password Store

This section describes the MQ Everyplace Password Store installation process.

MQ Everyplace Password Store (MQe Password Store) provides the function necessary to store user passwords into IBM WebSphere MQ Everyplace and transfer user password from MQ Everyplace to IBM Directory Integrator . The MQe Password Store package was created to support a growing number of IBM Directory Integrator plug-ins which intercept password changes on various product/platforms. Currently supported are IBM Directory Server, iPlanet (Sun ONE Directory Server) and Microsoft Active Directory. For example, the IBM Directory Server Password Synchronizer captures an IBM Directory Server user’s password change request. These catcher plug-ins can utilize the MQe Password Store function which facilitates the secure propagation of the change to the IBM Directory Integrator where it can be manipulated by an IBM Directory Integrator AssemblyLine.

The MQe Password Store package consists of two components:

**Storage Component**
Installed on the machine where the password synchronizer is installed and the password synchronizer is configured to use the Storage Component. For each intercepted password update, the Storage Component constructs a message containing the password data and sends it to MQe.

**MQe Connector**
Installed into the IBM Directory Integrator as a standard Connector. The MQe Connector is used to connect to MQe, retrieve and parse password update messages and feed an IBM Directory Integrator AssemblyLine.

MQe is actually embedded into the MQe Password Store components. The Storage Component contains an MQe QueueManager and the MQe Connector contains an MQe QueueManager. A remote connection between the two MQe QueueManagers transfers messages from the machine where the password synchronizer operates to the machine where the IBM Directory Integrator and the MQe Connector are run.

### Authenticated MQe Access

Tivoli Directory Integrator MQe components can be deployed to take advantage of MQe Mini-Certificate authenticated access. Use of certificate authenticated access prevents an anonymous MQe client Queue Manager and/or application submitting a change password request to the MQe Password Store Connector.

### Included files

The Password Store distribution package contains the following files:

**mqepwstore.jar**
Contains the Storage Component of the MQe Password Store

**mqeconfig.jar**
Component for automatic creation and configuration of the MQe QueueManagers

**mqeconfig.props**
Properties file for the configuration component

**ibmjms.jar**
Version 1.1 of IBM's interface definition for the JMS classes

**MQeBase.jar**
Contains MQ Everyplace base classes

**MQeJMS.jar**
Contains MQ Everyplace JMS support classes

**MQeSecurity.jar**
Contains MQ Everyplace secure queue access classes

**idipwcrypto.jar**
Contains the asymmetric encryption support for the MQe Password Store

**ibmjcefw.jar (Only installed with Domino Password Synchronizer)**
Used by the asymmetric encryption support

**ibmjceprovider.jar (Only installed with Domino Password Synchronizer)**
Used by the asymmetric encryption support

**ibmpkcs.jar (Only installed with Domino Password Synchronizer)**
Used by the asymmetric encryption support

**ibmpkcs11.jar (Only installed with Domino Password Synchronizer)**
Used by the asymmetric encryption support

**local_policy.jar (Only installed with Domino Password Synchronizer)**
Used by the asymmetric encryption support

**US_export_policy.jar (Only installed with Domino Password Synchronizer)**
Used by the asymmetric encryption support

**ibmjsse.jar (Only installed with Domino Password Synchronizer)**
Used by the SSL support.

**idicryptokeys.bat**
Utility bat file that creates a test/demo keystore file

**idicryptokeys.sh**
Utility shell script file that creates a test/demo keystore file

Installation of MQ Everyplace is not necessary. The two MQe components (**MQeBase.jar** and **MQeJMS.jar**) deployed by the installer are all that is needed to instantiate and use MQ Everyplace. However, if authenticated Queue Manager and queue deployments are required, separate installation of MQe is required.

## MQe Password Store Installation and Setup

This section details the steps required to install and configure the MQe Password Store.

### Storage Component installation and setup

Assume that the MQe Password Store is installed and integrated with the IBM Directory Server Password Synchronizer (the process for integration with the other password synchronizers is analogous).

If authenticated access is required for the MQe runtime and Tivoli Directory Integrator components, complete the configuration and startup of the MQe Mini-Certificate server before continuing with the steps described in this section. Ensure that certificates associated with ″PWStoreClient″ and ″PWStoreServer+passwords″ are available for issue.

The install folder of the IBM Directory Server Password Synchronizer is referred to as *IBM_Directory_Server_root*.

1. Checking the JAR file deployment.
   - Copy `mqeconfig.jar`, `mqeconfig.props` and `idicryptokeys.bat` (or `idicryptokeys.sh`) files into the *IBM_Directory_Server_root* folder.
   - For the Domino Password Synchronizer, the following JAR files are installed into the `*Domino_JVM_dir*/lib/ext` folder. For the other Password Synchronizers, they are installed in the `jars` subdirectory under the installation directory. For example, for the Sun One Directory Server Password Synchronizer, they are located in `*install_dir*/SunOne/jars` directory. The classpath is set via the appropriate `.props` file or registry entries to include these JAR files. Check the classpath entry for these JAR files:
     - mqepwstore.jar
     - ibmjms.jar
     - MQeBase.jar
     - MQeJMS.jar
     - MQeJMX.jar
     - MQeSecurity.jar
     - idipwcrypto.jar

     **Note:** On Domino 6.5.4, the following files must be copied from the `*domino_program_dir*\idipwsync\jce` folder to the `domino_JVM_dir/lib/ext` folder:
     - ibmjcefw.jar
     - ibmjceprovider.jar
     - ibmpkcs.jar
     - ibmpkcs11.jar
     - local_policy.jar
     - US_export_policy.jar
     - ibmjsse.jar

     Installation of MQ Everyplace is not necessary. The two MQe components copied (`MQeBase.jar` and MQeJMS.jar) are all that is needed to instantiate and use MQ Everyplace.
2. Create and configure the MQe QueueManager.

   The file `mqeconfig.jar` placed in *IBM_Directory_Server_root* contains a utility program (MQe Configuration Component) that automatically creates and configures the MQe QueueManager that is used by the Storage Component.
   - Before running the MQe Configuration Component, open its properties file **mqeconfig.props** and set values for the following properties:

     **clientRootFolder**
     The folder where you want to place the MQe QueueManager (for Windows, for example: `C:\\Program Files\\IBM\\IDSPasswordSynch\\MQePWStore`).

     **Note:** When specifying Windows filepaths in the property files, the backslash file separator ( \ ) must be escaped with a second backslash ( \\ ).

**serverIP**
: This is the IP address of the machine where the IBM Directory Integrator and the MQe Connector are deployed.

**communicationPort**
: The TCP/IP port that is used for communication between the two MQe QueueManagers.

**clientRegistryType**
: Optional. Required for authenticated MQe access deployments only. If used, value must be set to ″PrivateRegistry″. The Private Registry stores the certificates issued by the MQe Mini-Certificate server.

**clientRegistryPin**
: Optional. Required for authenticated MQe access deployments only. If used, this value represents the ″PIN″ access code used by the Tivoli Directory Integrator MQe Password Store to access the PrivateRegistry. This value will be stored as plain text in the result MQe ″.ini″ file produced by step ″b″ below.

**clientKeyRingPassword**
: Optional. Required for authenticated MQe access deployments only. This value is used when requesting a certificate from the MQe Mini-Certificate server It is the seed value for certificate generation. This value will be stored as plain text in the result MQe ″.ini″ file produced by step ″b″ below.

**certServerReqPin**
: Optional. Required for authenticated MQe access deployments only. This value is used as a one time authentication PIN by this Queue Manager when requesting certificates from the MQe Mini-Certificate server. This value must match the ″Request PIN″ value from Mini-Certificate server setup steps 4 and 6.

**certServerIPAndPort**
: Optional. Required for authenticated MQe access deployments only. This value is used as the destination address for MQe Mini-Certificate server requests. The format of the value is ″FastNetwork:<host>:<port>″, where host must be the machine name or TCP IP address where the MQe Mini-Certificate server is running, and port value must match the ″Port″ value from Mini-Certificate server setup step 3 .

**debug**
: Specify **true** or **false** to correspondingly turn debug information on or off.

The following is a sample **mqeconfig.props** configuration file:

```
clientRootFolder=C:\\Program Files\\IBM\\IDSPasswordSynch\\MQePWStore
serverRootFolder=C:\\Program Files\\IBM\\IBMDirectoryIntegrator\\MQePWStore
serverIP=127.0.0.1
communicationPort=41001

##
## Uncomment the following lines if authenticated MQe access is required
##
#clientRegistryType=PrivateRegistry
#clientRegistryPin=<Private client registry access PIN>
#clientKeyRingPassword=<Seed value for certificate generation>
#certServerReqPin=<One time certificate request PIN>
```

```
#certServerIPAndPort=FastNetwork:<Mini-Certificate server
     hostname or IP>:<port>

debug=true
```

**Note:** When specifying Windows filepaths in the property files, the backslash file separator ( \ ) must be escaped with a second backslash ( \\ ).

The **serverRootFolder** property is not used when configuring the Storage Component (it is used to configure the QueueManager at the MQe Connector) and its value is not taken into account here.

- To create and automatically configure MQe QueueManager for the Storage Component, open a command prompt in the *<IBM_Directory_Server_root>* folder and enter the following command (as one line):

```
jvm\jre\bin\java -cp "./mqeconfig.jar"
    com.ibm.di.plugin.mqe.config.MQeConfig mqeconfig.props create client
```

The log of this command is displayed on the console. After successful completion, the message Client MQe configuration successfully completed displays. If the **mqeconfig.props** file contains the optional parameters for MQe authenticated access, this step will automatically request the necessary certificates from the MQe Mini-Certificate server.

Tip: If attempting to perform an MQe certificate authenticated access deployment, it is important to remember that certificates may be requested once only per authenticate-able entity. If an exception message similar to the one below is reported during configuration, it may be necessary to re-enable certificate issue for that entity using the Mini-Certificate server GUI.

```
[MQeConfig] [28/07/05 10:10:01]: Action failed:
Code=351;com.ibm.mqe.MQeException: Registration exception =
com.ibm.mqe.MQeException: certificate request failed[PWStoreClient 4]
 (code=8)[PWStoreClient 8] (code=351) [MQeConfig] [28/07/05 10:10:01]:
Error: Server MQe configiration failed; exception:java.lang.Exception:
Code=351;com.ibm.mqe.MQeException: Registration exception = com.ibm.mqe.
MQeException: certificate request failed[PWStoreClient4] (code=8)
[PWStoreClient 8] (code=351)
```

**Note:** If you need to change the configuration of the QueueManager, you have two options:

- Delete the QueueManager from the disk and create it again following the previous procedure, or
- Install an MQ Everyplace admin tool compatible with MQ Everyplace 2.0.2.5 QueueManagers (for example, MQe Explorer) and use it to change the QueueManager settings.

3. Integrate with the Password Synchronizer.
   a. To integrate the MQe Password Store in the IBM Directory Server Password Synchronizer, open the configuration file of the IBM Directory Server Password Synchronizer (`IBM_Directory_Server_root/idspwconfigWin.props`) and edit the **syncClassName** property:

   ```
   syncClassName=com.ibm.di.plugin.mqe.store.MQePasswordStore
   ```

   b. To integrate the MQe Password Store in the Microsoft Active Directory Password Synchronizer, the class `com.ibm.di.plugin.mqe.store.MQeNTPasswordStore` must be specified when installing the Microsoft Active Directory Password Synchronizer.
4. Configure the Storage Component.

   The configuration file of the Storage Component is called **mqepwstore.props** and must be placed in a folder that is on the CLASSPATH of the Password Synchronizer (the classpath of the IBM Directory Server Password Synchronizer is specified through the jvmClassPath property in the IBM Directory Server Password Synchronizer configuration file). Store the **mqepwstore.props** file in the *<IBM_Directory_Server_root>* folder.

a. Use the following command from the *<IBM_Directory_Server_root>* folder to generate a partial properties file with some basic settings:

```
jvm\jre\bin\java com.ibm.di.plugin.mqe.store.MQeGenConfigFile
       <encryptKeyStoreFilePassword>
```

where *<encryptKeyStoreFilePassword>* is the un-encoded password for client keystore used for encrypting passwords (for example, **secret**). The file **mqepwstore.props** is generated with the following content:

```
#MQe Password Store Properties (with encoded passwords)
#Thu Apr 03 14:32:02 EEST 2003
debug=false
logFile=
encryptKeyStoreFilePassword=0c0bf0e3146b
encryptKeyStoreCertificate=
encryptKeyStoreFilePath=
notificationPort=41002
qmIniFileName=
encrypt=true
```

b. This utility does not set all the required properties. It sets the encrypted password and some default property values, but you must set most of the properties:

**qmIniFileName**
The path to the .ini file of the generated MQe QueueManager (usually `C:\\Program Files\\IBM\\IDSPasswordSynch \\MQePWStore\\pwstore_client.ini`).

**notificationPort**
The TCP/IP port that is used when the MQe Connector sends notifications to the Storage Component. Default value is **41002**.

**logFile**
The file path of the Storage Component log file. For example, `C:\\Program Files\\IBM\\IDSPasswordSynch\\mqestore.log`.

**encrypt**
Specify **true** or **false** to correspondingly turn the encryption of passwords on or off.

**encryptKeyStoreFilePath**
The path of the JKS file that is used to encrypt passwords (only taken into account when encrypt is set to **true**).

**encryptKeyStoreFilePassword**
The encrypted password of the JKS file (only taken into account when encrypt is set to **true**). This maps to the `-storepass` parameter for keytool -genkey (see `idicryptokeys.bat` or `idicryptokeys.sh`).

**encryptKeyStoreCertificate**
The alias of the key from the JKS file.

**debug** Specify **true** or **false** to correspondingly turn debug information on or off.

c. When you choose to store the passwords in encrypted format (encrypt set to **true**), password data is encrypted or decrypted using the RSA key algorithm. You can use the provided `idicryptokeys.bat` (or `idicryptokeys.sh`) to help build a test keystore for testing. If you use the `idicryptokeys.bat` to create a test keystore, your values in the properties file are as follows:

- **encryptKeyStoreFilePath**=***anypath***/idicryptotest.jks where *anypath* is the location where you keep the JKS file created by `idicryptokeys.bat` (see `-keystore` parameter setting)
- **encryptKeyStoreFilePassword**=*encrypted value of secret* (see `-storepass` parameter setting)
- **encryptKeyStoreCertifcate**=idicryptotest (see `-alias` parameter setting)

d. If you choose to code the **mqepwstore.props** properties file, or wish to later change the *encryptKeyStoreFilePassword*, you can use a second utility which encodes your new password, so you can copy it into the properties file. To obtain an encoded version of your password issue the following command from the *IBM_Directory_Server_root* folder:

```
jvm\jre\bin\java com.ibm.di.plugin.idipwsync.EncodePW password
```

where *password* is your ASCII password (for example, secret). For example:

```
jvm\jre\bin\java com.ibm.di.plugin.idipwsync.EncodePW secret
```

returns:

```
0c0bf0e3146b
```

The following is an example of a completed **mqepwstore.props** file:

```
#MQe Password Store Properties (with encoded passwords)
#Thu Apr 03 14:32:02 EEST 2003
qmIniFileName=C:\\Program Files\\IBM\\IDSPasswordSynch\\MQePWStore\\

pwstore_client.ini
notificationPort=41002
logFile=C:\\Program Files\\IBM\\IDSPasswordSynch\\mqestore.log
encrypt=true
encryptKeyStoreFilePassword=0c0bf0e3146b
encryptKeyStoreFilePath=C:\\Program Files\\IBM\\IDSPasswordSynch\\

idicryptotest.jks
encryptKeyStoreCertificate=idicryptotest
debug=true
```

## MQe Connector Installation and Setup

**Included files:** The distribution package installs the following files to the specified subdirectories under the IBM Directory Integrator root location:

**\jars\connectors\MQePasswordStoreConnector.jar**
The MQe Password Store Connector package.

**\jars\plugins\mqeconfig.jar**
Component for automatic creation and configuration of the MQe QueueManagers.

**\jars\plugins\mqeconfig.props**
Properties file for the configuration component.

**\jars\plugins\mqeconfig.bat**
Utility bat file for the configuration component that sets the classpath to the IBM Directory Integrator directory structure

**/jars/plugins/mqeconfig.sh**
Utility shell script file for the configuration component that sets the classpath to the IBM Directory Integrator directory structure

**\jars\ibmjms.jar**
Version 1.1 of IBM's interface definition for the JMS classes

**\jars\MQeBase.jar**
Contains MQ Everyplace base classes

**\jars\MQeJMS.jar**
Contains MQ Everyplace JMS support classes

**\jars\MQeSecurity.jar**
Contains MQ Everyplace security classes

**Installation instructions:**

1. Deploy the jar files.

   The install folder of the IBM Directory Integrator is referred to as *IBM_Directory_Integrator_root*.

   a. Copy the `MQeConnector.jar` file in the *`IBM_Directory_Integrator_root`*`/jars/connectors` folder.
   b. Copy the files `mqeconfig.jar`, `mqeconfig.props` and `mqeconfig.bat` (or `mqeconfig.sh`) into the *`IBM_Directory_Integrator_root`* folder.
   c. Copy the files `MQeBase.jar`, `MQeJMS.jar` and `jms.jar` into the *`IBM_Directory_Integrator_root`*`/jars` folder. Installation of MQ Everyplace is not necessary. The two MQe components copied - `MQeBase.jar` and `MQeJMS.jar` - are all that is needed to instantiate and use MQ Everyplace.

2. Create and configure the MQe QueueManager.

   The file `mqeconfig.jar` placed in *`IBM_Directory_Integrator_root`* contains a utility program (MQe Configuration Component) that automatically creates and configures the MQe QueueManager that is used by the MQe Connector.

   If authenticated access is required for the MQe runtime and Tivoli Directory Integrator components, complete the configuration and startup of the MQe Mini-Certificate server before continuing with the steps described in this section. Ensure that certificates associated with ″PWStoreServer″ and ″PWStoreServer+passwords″ are available for issue.

   Before running the MQe Configuration Component, open the properties file mqeconfig.props and set values for the following properties:

   **serverRootFolder**
   The folder where you want to place the MQe QueueManager (for example, `C:\\Program Files\\IBM\\IBMDirectoryIntegrator\\MQePWStore`).

   **communicationPort**
   The TCP/IP port that is used for communication between the two MQe QueueManagers (make sure to specify the same port that was used to configure the Storage Component MQe QueueManager).

   **serverRegistryType**
   Optional. Required for authenticated MQe access deployments only. If used, value must be set to ″PrivateRegistry″. The Private Registry stores the certificates issued by the MQe Mini-Certificate server.

   **serverRegistryPin**
   Optional. Required for authenticated MQe access deployments only. If used, this value represents the ″PIN″ access code used by the Tivoli Directory Integrator MQe Password Connector to access the PrivateRegistry. This value will be stored as plain text in the result MQe ″.ini″ file produced by step ″b″ below.

**serverKeyRingPassword**
Optional. Required for authenticated MQe access deployments only. This value is used when requesting a certificate from the MQe Mini-Certificate server. It is the seed value for certificate generation. This value will be stored as plain text in the result MQe ″.ini″ file produced by step ″b″ below.

**certServerReqPin**
Optional. Required for authenticated MQe access deployments only. This value is used as a one time authentication PIN by this Queue Manager when requesting certificates from the MQe Mini-Certificate server. This value must match the ″Request PIN″ value from Mini-Certificate server setup steps 4 and 6.

**certServerIPAndPort**
Optional. Required for authenticated MQe access deployments only. This value is used as the destination address for MQe Mini-Certificate server requests. The format of the value is ″FastNetwork:<host>:<port>″, where host must be the machine name or TCP IP address where the MQe Mini-Certificate server is running, and port value must match the ″Port″ value from Mini-Certificate server setup step 3.

**debug** Specify true or false to correspondingly turn debug information on or off.

The following is a sample mqeconfig.props configuration file:

```
clientRootFolder=C:\\Program Files\\IBM\\IDSPasswordSynch\\MQePWStore
serverRootFolder=C:\\Program Files\\IBM\\IBMDirectoryIntegrator\\MQePWStore
serverIP=127.0.0.1
communicationPort=41001

##
## Uncomment the following lines if authenticated MQe access is required
##
#serverRegistryType=PrivateRegistry
#serverRegistryPin=<Private Registry access PIN>
#serverKeyRingPassword=<Private Registry key>
#certServerReqPin=<certificate request PIN>
#certServerIPAndPort=FastNetwork:<host>:<port>

debug=true
```

The **clientRootFolder** and **serverIP** properties are not used when we configure the QueueManager of the MQe Connector (they are only used to configure the QueueManager of the Storage Component) and their values are not taken into account here.

To create and automatically configure MQe QueueManager for the MQe Connector, open a command prompt in the *IBM_Directory_Integrator_root* folder and execute the following command:

```
mqeconfig.bat mqeconfig.props create server
```

The log of this command is displayed on the console. After successful completion, the message Server MQe configuration successfully completed is displayed. If the mqeconfig.props file contains the optional parameters for MQe authenticated access, this step will automatically request the necessary certificates from the MQe Mini-Certificate server.

Tip: If attempting to perform an MQe certificate authenticated access deployment, it is important to remember that certificates may be requested once only per authenticate-able entity. If an exception message similar to the one below is reported during configuration, it may be necessary to re-enable certificate issue for that entity using the Mini-Certificate server GUI.

```
[MQeConfig] [28/07/05 10:10:01]: Action failed: Code=351;com.ibm.
mqe.MQeExceptio n: Registration exception = com.ibm.mqe.
MQeException: certificate request failed [PWStoreServer 4]
(code=8)[PWStoreServer 8] (code=351) [MQeConfig] [28/07/05
10:10:01]: Error: Server MQe configiration failed; exception:
java.lang.Exception: Code=351;com.ibm.mqe.MQeException:
Registration exception = com.ibm.mqe.MQeException: certificate
 request failed[PWStoreserver 4] (code=8)[PWStoreServer 8]
(code=351)
```

**Note:** If for some reason you need to change the configuration of the QueueManager, you have two options:

- Delete the QueueManager from the disk and create it again following the previous procedure.
- Install an MQ Everyplace admin tool (for example, MQe Explorer) compatible with MQ Everyplace v2.0.2.5 QueueManagers and use it to change the QueueManager settings.

3. Configure the MQe Connector.

   The MQe Connector is configured through the IBM Directory Integrator as a standard IBM Directory Integrator Connector.

   The following parameters are available:

   **GetNext Timeout**
   Specify the number of milliseconds the Connector waits for a new password update message to appear in the QueueManager queue. Specify **-1** to wait forever, and **0** to return immediately if no message is available.

   **Storage notification server**
   Specify in a <*host*>:<*port*> port format the Storage Component server that listens for notifications from the MQe Connector. The default value for the port is 41002 and the host must be the IP address of the machine where the Password Synchronizer and the Storage Component are deployed.

   **Decrypt messages**
   Check this field if the Storage Component encrypts the password update messages and they need to be decrypted by the MQe Connector.

   **Key Store File**
   The path of the JKS file used to decrypt password data (only taken into account when the **Decrypt messages** field is selected).

   **Key Store File Password**
   The password of the JKS file (only taken into account when the **Decrypt messages** field is selected).

   **Key Store Certificate Alias**
   The alias of the key from JKS file (only taken into account when the **Decrypt messages** field is selected).

   **Key Store Certificate Password**
   The password used to retrieve the private key. If not specified the <*KeyStoreFilePassword*> is used to retrieve the private key (only taken into account when the **Decrypt messages** field is selected).

**Detailed Log**
Check this field for more detailed log messages.

When the MQe Connector is configured to decrypt password data (the **Decrypt messages** field is selected), use the JKS file generated for the Storage Component and specify the corresponding values for *KeyStoreFilePassword*, *KeyStoreCertificateAlias* and *KeyStoreCertificatePassword*.

To enable the Tivoli Directory Integrator server to start the MQe Queue Manager, go to the **Global-Properties** and set the **systemqueue.on** property to *true*.

4. MQe Connector schema.

   MQe Connector supports the Iterator mode only. It retrieves password update messages from MQe, parses them and constructs Entry objects using a fixed schema of Attributes and Properties.

   Each Entry delivered by the MQe Connector is populated with the following Attributes and Properties:

   **UserId (attribute)**
   Contains the ID of the user (entry) for which the password has been updated. In the case of LDAP Password Synchronizers (IBM Directory Server and Sun Directory Server), this attribute specifies the distinguished name of the LDAP entry.

   **UpdateType (attribute)**
   Contains one of the following values:

   **replace**
   Specifies that the entry's list of passwords has been replaced with the values specified by the Passwords attribute.

   **add** Specifies that the values of the Passwords attribute have been added to the entry's list of passwords.

   **delete** Specifies that the values of the Passwords attribute have been removed from the entry's list of password.

   **Passwords (attribute)**
   A multi-valued attribute containing the changed password values. This attribute is always present in the Entry object, even when the number of password values is zero.

   **PasswordCount (property)**
   Specifies the number of password values contained in the Passwords attribute. The same meta-information can be retrieved through the Passwords attribute object.

   **Note:** This is a Property object which is not mapped in the AssemblyLine Attribute Mapping process. You can access this Property through the **conn** Connector object only.

**Verifying installation of the MQe QueueManagers:** The MQe Configuration Component delivered in `mqeconfig.jar` can be run in test mode to verify the installation of the two MQe QueueManagers and test the communication channel between them.

To run the test, do the following:

1. Open a command prompt on the IBM Directory Integrator machine, in the *`IBM_Directory_Integrator_root`* folder. Enter the following command:

```
mqeconfig.bat mqeconfig.props test server
```

The message **Press Enter to receive test message ...** is displayed on the console. Do not press anything.

2. Open a command prompt on the Storage Component machine, in the *IBM_Directory_Server_root* folder. Enter the following command (as one line):

   ```
   jvm\jre\bin\java -cp "./mqeconfig.jar" com.ibm.di.plugin.mqe.config.MQeConfig
         mqeconfig.props test client
   ```
3. The message **Press Enter to send test message ...** is displayed on the console. Press **Enter**.
4. The message **Test message sent.** is displayed, followed by a message **Press Enter to close ....** Press **Enter** again.

   The message **QueueManager terminated.** indicates clean termination of the QueueManager and the application exits.
5. Go back to the first console, on the IBM Directory Integrator machine.
6. Press **Enter**. A message **Success: test MQe message successfully received.** indicates that the two QueueManagers are properly installed and configured to communicate with each other. A message starting with **Test failed:** indicates that the QueueManagers are not properly installed or configured.
7. When the message **Press Enter to close...** displays, press **Enter**. A message **QueueManager terminated.** indicates clean termination of the QueueManager and the application exits.

## Installation and Configuration of MQe for PKCS7

In Tivoli Directory Integrator version 6.1.1 (and onwards), the MQe Password Store and the MQe Password Store Connector support PKCS7. Both signing and encryption are supported. The use of PKCS7 encapsulation is optional (it is turned off by default).

When using PKCS7, both the MQe Password Store and the MQe Password Store Connector must be configured to use PKCS7. If only one is configured to use PKCS7, an error will occur.

**Certificates used**
: PKCS7 involves both signing and encryption. Both signing and encryption need certificates in order to function.

**Signing**
: Signing is used to verify that the sender of the message is the one he or she claims to be. In this particular scenario, the MQe Password Store Connector needs to verify that the sender of a password change notification message is actually a trusted MQe Password Store. This is achieved by having the password store use the certificate's private key. Then the Connector uses the password store's public key to verify that signature. It is possible to have several password stores sending messages to a single MQe Password Store Connector. In this case the Connector must be configured so that its .jks file contains the public keys of each of the trusted password stores.

**Encryption**
: Encryption is achieved by having the password store use the public key of the Connector to encrypt the message. Then the Connector uses its private key to decrypt the message.

**Certificate management**

Both the Connector and Password Store use .jks files to store certificates. To use PKCS7 for the MQe Password Store, the .jks file must contain the public key of the Connector as a trusted signer certificate, and the private-public key pair of the password store. To use PKCS7 for the Connector, the .jks file must contain the private-public key pair of the Connector , and the public key of each trusted password store as a trusted signer certificate. The **ikeyman** tool is used to handle the JKS files. For information on how to create these .jks files, see "Using ikeyman for JKS Files" on page 101.

**How to enable and disable PKCS7:** A new boolean property is introduced, which indicates whether the PKCS7 encapsulation option is turned on or off. It is defined in the **MQePasswordStore** properties file. If it is set to *TRUE* the other parameters related to PKCS7 must also be given an appropriate value (additional information about these properties is supplied below).

The **pkcs7** parameter must be set to *true* if the **pkcs7** option is activated at the other end (that is, at the **MqePasswordStoreConnector**). Usually the **MqePasswordStoreConnector** must be configured according to the **MqePasswordStore** properties, but supporting multiple plug-ins raises issues where different values are set for the **pkcs7** parameter of each plugin. The pre-Tivoli Directory Integrator 6.1.1 PKI encryption and **pkcs7** encapsulation cannot be used simultaneously because PKCS7 is able to encrypt messages by itself. The public key of the signing certificate is sent to the **MqePasswordStoreConnector** as part of the message and is used by the Connector to verify the plugin.

The implementation relies on the standard Java packages, `java.security` and `java.security.cert`, as well as the IBM packages `com.ibm.security.pkcs7`, `com.ibm.security.pkcsutil` and `com.ibm.misc`. After successful initialization of the PKCS7 parameter (mPkcs7), in case of *true*, the other parameters from the props file related to this option are as follows:

- the path to the JKS file,
- the alias name of the MQePasswordStore's certificate,
- the alias name of the MQePasswordStoreConnector's certificate, and
- the password to JKS file.

A key store is created from the supplied JKS as well as two X509Certificate instances, which represent the certificate of MQePasswordStore (mMqeStoreCert) and MQePasswordStoreConnector (mMqeConnectorCert). For the signing of the emitted messages, the private key of the first certificate (mMqeStorePrivateKey) also needs to be extracted. The implementation does not support extracting a key with a password (the existing password of the key store is used instead). Therefore, either do not set the password to the self-signed certificates in the JKS file, or make it the same as the password in the key store.

The properties file `mqeconfig.props` includes five new parameters, which must be configured before using the PKCS functionality. After the desired plugin is installed and configured to work with MQePasswordStore, locate the `mqepwstore.props` file in the plugin installation directory. The five new parameters are as follows:

**pkcs7=[true | false]**
This indicates whether or not the option is turned on.

**pkcs7KeyStoreFilePath=[filePath\\myjks.jks]**
The file path and the name of the JKS file.

**pkcs7KeyStoreFilePassword=[password]**
The password for the JKS file.

**Note:** The encrypted version of the password must be written, not the actual one.

**pkcs7MqeStoreCertificateAlias=[alias]**
The alias of the MQePasswordStore's certificate.

**pkcs7MqeConnectorCertificateAlias=[alias]**
The alias of the MQePasswordStoreConnector's certificate.

To use the PKCS7 encapsulation, set the **pkcs7** parameter to *true*. The other above-mentioned parameters are not considered when **pkcs7** is set to *false*.

The **pkcs7KeyStoreFilePath** parameter must provide the full absolute path to the MQePasswordStore's .jks file, along with its name. Double slash ″\\″ must be used instead of single ″\″, when specifying the file path on Windows platforms.

For security reasons, the parameter **pkcs7KeyStoreFilePassword** accepts the encrypted version of the JKS file password, not the actual one. The encrypted password string can be generated by using the **encodepw** command line utility, which is bundled with the password store and stored in the `com.ibm.di.plugin.idipwsync.EncodePW` Java class. After you have set the Java class path, the **encodepw** utility can be started with the following command line:

```
java com.ibm.di.plugin.idipwsync.EncodePW password
```

See the plug-ins documentation for more information about the **encodepw** utility. Both **pkcs7MqeStoreCertificate** and **pkcs7MqeConnectorCertificate** must be given the alias of the corresponding certificate as a value. Below is an example of such a configuration.

*Example:* For this example, the name of the .jks file used to store the certificates is `mqepkcs7.jks` and the path is `C:\Program Files\IBM\DiPlugins\`. The .jks file must contain both the self-signed certificate of the MQePasswordStore and the trusted signer certificate of the MQePasswordStoreConnector. The password is secret and using the utility **encodepw** (as described above) we obtain the encrypted version: *0c0bf0e3146b*. The alias of theMQePasswordStore in this situation is *mqestore* and the alias of the theMQePasswordStoreConnector is *mqeconn*.

```
pkcs7=true
pkcs7KeyStoreFilePath=C:\\Program Files\\IBM\\DiPlugins\\mqepkcs7.jks
pkcs7KeyStoreFilePassword=0c0bf0e3146b
pkcs7MqeStoreCertificateAlias=mqestore
pkcs7MqeConnectorCertificateAlias=mqeconn
```

**Notes:**

1. The **pkcs7KeyStoreFilePath** parameter must supply the full absolute path to the **MQePasswordStore**'s JKS file along with its name. Double slash "\\" must be used instead of single "\", when specifying the file path on Windows platforms. The parameter **pkcs7KeyStoreFilePassword** accepts the encrypted version of the JKS file password, not the actual one, because of security reasons. The encrypted password string can be generated by using the **encodepw** command line utility (stored in the

`com.ibm.di.plugin.idipwsync.EncodePW` Java class), which is bundled with the password store. Having set the Java class path, this utility can be started with the following command line:

```
java com.ibm.di.plugin.idipwsync.EncodePW yourpassword
```

For more information on this utility, see the plug-ins documentation.

2. Both **pkcs7MqeStoreCertificate** and **pkcs7MqeConnectorCertificate** must be given the alias of the corresponding certificate as a value.

**MQePasswordStore Connector for PKCS7:** The MQePasswordStore Connector also uses a JKS file as a key/trust store. With the PKCS7 option activated, it verifies the signature of each received message by comparing the Signer certificate with the ones in its trust store. In case of a match it verifies the message signature. If the signature verification is successful, the Connector accepts the message and decrypts it with the Connector's private key from its own certificate. The implementation relies on the standard Java packages, `java.security` and `java.security.cert`, as well as the IBM packages `com.ibm.security.pkcs7`, `com.ibm.security.pkcsutil` and `com.ibm.misc`.

**How to enable and disable PKCS7 at the Connector**

The new Boolean configuration parameter available in the Connector GUI – **usePkcs7** – indicates whether the received messages are PKCS7 signed and takes care of the control over the decryption. It is important to use the same value of the **pkcs7** parameter at both ends (**MQePasswordStore** and **MQePasswordStoreConnector**) otherwise the password changes will not be synchronized if errors occur. In Tivoli Directory Integrator 6.1.1, this parameter is represented by a checkbox. When it is checked, PKCS7 is used. In this situation, the other parameters related to the option must also be given the correct values. See below for more information about the parameters `pkcs7KeyStoreFilePath`, `pkcs7KeyStoreFilePassword`, and `pkcs7MqeConnectorCertAlias`.

**Multiple plugin support**

The MQePasswordStore Connector supports receiving messages from multiple password stores. When PKCS7 encapsulation is activated for the Connector, all the password stores must use the PKCS7 option (this would make the stores sign their messages with their certificates and encrypt them with the certificate of the receiving MQePasswordStore Connector). Afterwards the received messages must be verified by the Connector using the public key of the password stores' certificate, which is sent as part of the message. For this purpose, the MQePasswordStore Connector must be supplied with a JKS file containing the plug-in certificates. For more information about working with JKS files, see "Using ikeyman for JKS Files" on page 101.

**Certificate management**

As with the MQePasswordStore, a JKS file is required in order to start working with the PKCS7 functionality. It must contain not only MQePasswordStoreConnector's certificate, but also the certificates of all the password stores, which send messages to it. The MQePasswordStore Connector's certificate is a self-signed personal certificate, whose private key is used to decrypt the messages from the password store. The password stores' certificates are trusted signer certificates, which are supplied from the MQePasswordStore's JKS file of each store. Every received message is then verified, as the public key attached to it is compared with the available private keys in the JKS file. In the case of a match, the message signature is verified against the certificate and then the message is decrypted using the Connector's own private key.

To configure the MQePasswordStore Connector, appropriate values must be given to the new Connector Configuration parameters related to the use of PKCS7 encapsulation.

**'usePkcs7' parameter**
This Boolean parameter indicates whether PKCS7 encapsulation is used or not. The default value is *false*. All other parameters related to the PKCS7 functionality are considered if only this parameter is set to *true*.

| | |
|---|---|
| label | PKCS7 |
| description | Defines whether the Pkcs7 signing and encryption is used |
| type | Boolean. Available values are "true" and "false" |
| default value | False |

**'pkcs7KeyStoreFilePath' parameter**
This parameter must be contain the full name and path of the MQePasswordStoreConnector's JKS file. When specifying the file path on Windows platforms, there is no need of a double slash "\\" instead of single "\" .

| | |
|---|---|
| label | PKCS7 Key Store File |
| description | Defines path and name of the MQePasswordStoreConnector's JKS file |
| type | String |
| default value | None |

**'pkcs7KeyStoreFilePassword' parameter**
This parameter accepts the actual value of the password for the JKS file (whereas the encrypted version is required for the MQePasswordStore's property).

| | |
|---|---|
| label | PKCS7 Key Store File Password |
| description | Defines MQePasswordStoreConnector's JKS Password |
| type | Password |
| default value | None |

**'pkcs7MqeConnectorCertAlias' parameter**
Similar to MQePasswordStore, this parameter accepts the alias of the MQePasswordStoreConnector's certificate as it is saved in the .jks file, without any extensions.

| | |
|---|---|
| label | MQeConnector Certificate Alias |
| description | Defines the name of the MQePasswordStoreConnector certificate alias |
| type | String |
| default value | None |

**Using ikeyman for JKS Files:** The **ikeyman** tool is available with every JVM distributed with Tivoli Directory Integrator, and can be found in the JVM intall sub-directory. For example on a Tivoli Directory Integrator install on Windows, the ikeyman executable is located in *install_dir*`\jvm\jre\bin` directory, where *install_dir* is the installed directory of the Tivoli Directory Integrator.

Follow the steps below to create the required keystore and truststore .jks files:

**Create a JKS file**

1. Click **Key Database File** then **New** and choose **JKS** together with the desired name and file path.
2. Enter a password and make a note of it (it must be provided later when configuring the components).

You will need to create at least two JKS files, one for the MQePasswordStore and another one for the MQePasswordStoreConnector. Additional JKS files will be required if you have more than one plug-in.

**Create a certificate**

To create a new certificate click on the dropdown menu above the list of certificates and choose **Personal Certificates**. Then click on **New Self-Signed** and enter the appropriate information.

**Transfer certificates**

Add the just-created self-signed certificates from the MQePasswordStore's JKS file to the MQePasswordStoreConnector's JKS file, and vice versa. To do this, extract the certificate as DER binary data:

1. Click on **Extract Certificate** and choose **Data Type** then **DER Binary data**.
2. Save to an appropriate location with the desired name and open the other JKS file.
3. Choose the Signer Certificates list, then click **Add**.
4. Locate the file with the DER extracted data and click **OK**.

**Note:** The implementation of PKCS7 in Tivoli Directory Integrator 6.1.1 does not support certificates, which are secured with an additional password (except the one set for the JKS file).

## Installation and Configuration of MQe Mini-Certificate Server

If secure authenticated access is required for the MQe runtime and Tivoli Directory Integrator MQe components, then it will be necessary to install and configure the MQe Mini-Certificate Server. This MQe component is available in MQe Server Support ES06, and must be downloaded separately. The Mini-Certificate server must be running on a machine within the target network, and be accessible by both the Tivoli Directory Integrator MQe Storage Component and Tivoli Directory Integrator MQe Password Connector machines. Accessibility is required during setup of the MQe Queue Manager for each of these Tivoli Directory Integrator components, and is not necessary at runtime. Use of MQe Mini-Certificates prevents anonymous MQe applications from submitting or processing change password MQe messages.

The role of the Mini-Certificate server is to issue certificates to each of the MQe Queue Managers associated with the Tivoli Directory Integrator Storage Component and Tivoli Directory Integrator MQe Password Connector. During configuration, the Queue Manager request certificates from the MQe Mini-Certificate server. The Mini-Certificate server issues certificates to the Queue Managers only when the Queue Manager sends the correct request ″PIN″ value. The Queue Managers securely store the certificates in their local Private Registries.

At runtime, the certificates are presented and verified before change password requests are submitted and received by the Tivoli Directory Integrator MQe components.

**Setup:** A file named `MQe_MiniCertServer.pdf` ships and installs with MQe. Please refer to this manual for specific detailed instructions.

The goal of configuring the Mini-Certificate server is to create the ″authenticate-able entities″ permitted to use the MQe runtime and associated certificates. Authenticate-able entities will be created for the Tivoli Directory Integrator MQe client Queue Manager, Tivoli Directory Integrator server Queue Manager, and the primary queue which transfers the password change notification messages.

A summary specific to Tivoli Directory Integrator is supplied below:

1. Install the MQe kit on machine in the target network. The machine must accessible from the machine running the Tivoli Directory Integrator MQe components. It is possible to install MQe on one of the Tivoli Directory Integrator component host machines if desired.
2. Open the MQe Mini-Certificate GUI.
3. Create a new profile. On the note pane window that results, configure the following:
   - GENERAL

     **Name** The name of the profile. For example: ″TDIMiniCertificateServer″

     **Passphrase/Confirm PP**
     The password to protect access and modification to this profile.
   - COMMS

     **Port** The TCP port that the Mini-Certificate server will listen for incoming certificate issue requests from the Tivoli Directory Integrator Queue Managers.

     **Adapter**
     Set value to ″FastNetwork″.
   - STORAGE

     **Adapter**
     Set value to ″RegistryAdapter″

     **Path** The directory path root below which, MQe will create the Mini-Certificate server registry structure.
4. Create the Tivoli Directory Integrator MQe Client Queue Manager entity. With the new profile created, it is now possible to create the required authenticate-able entities. Right click on the ″MQe root″ node and select ″New Entity″. On the note pane window that results, configure the following:
   - GENERAL

     **QMgr** Set the value to ″PWStoreClient″.

     **EnableIssue**
     Check the box

     **Request PIN**
     The PIN used by the Tivoli Directory Integrator Storage Component Queue Manager to request certificates. This value must match the value set for ″certServerReqPin″ in the `mqeconfig.props` file.

5. Create the Tivoli Directory Integrator MQe Server Queue Manager entity. Right click on the ″MQe root″ node and select ″New Entity″. On the note pane window that results, configure the following:
   - GENERAL

     **QMgr** Set the value to ″PWStoreServer″.

     **EnableIssue**
     Check the box

     **Request PIN**
     The PIN used by the Tivoli Directory Integrator Storage Component Queue Manager to request certificates. This value must match the value set for ″certServerReqPin″ in the `mqeconfig.props` file.
6. Create the password queue entity. Right click on the ″PWStoreServer″ node and select ″New Entity″. On the note pane window that results, configure the following:
   - GENERAL

     **QMgr** Set the value to ″PWStoreServer″.

     **Queue**
     Set the value to ″passwords″.

     **EnableIssue**
     Check the box

     **Request PIN**
     The PIN used by the Tivoli Directory Integrator Storage Component Queue Manager to request certificates. This value must match the value set for ″certServerReqPin″ in the `mqeconfig.props` file, and the same as that used in steps 4 and 5 above.
7. Start the server.

## Issuing Certificates for Multiple Client side Instances - Tivoli Directory Integrator MQe Password Store

In some deployments, it may be necessary to configure multiple Tivoli Directory Integrator MQe Password Store components.

For example, if password change plug-ins have been configured for multiple Windows Domain Controllers. In this case, then it is likely that there will separate instances of MQe client side Queue Managers with the name ″PWStoreClient″.

Additionally, for each of the client Queue Managers, there will be a remote queue proxy connection to the MQe server side Queue Manager queue used by the Tivoli Directory Integrator MQe Password Connector. The remote queue proxy name is ″PWStoreServer+passwords″.

When this type of deployment scenario is used, the authentication certificates associated with these two MQe entities (that is, ″PWStoreClient″ and ″PWStoreServer+passwords″) will be requested and issued multiple times. This happens each time the **mqeconfig** utility is executed as described in Storage Component Installation and Setup.

Before executing the second and each subsequent instances of the **mqeconfig** utility, it will necessary to re-enable certificate issue for each of the MQe entities mentioned above. The steps for this are described below:

1. Open the MQe Mini-Certificate GUI.

2. Select the Tivoli Directory Integrator MQe Client Queue Manager entity. Right click on the ″PWStoreClient″ node and select ″Properties″. On the note pane window that results, configure the following:
   - GENERAL

     **Enable Issue**
       Check the box.

     **Request PIN**
       The PIN used by the Tivoli Directory Integrator Storage Component Queue Manager to request certificate. This value must match the value for ″certServerReqPin″ in the `mqeconfig.props` file where the **mqeconfig** utility will execute.
3. Select the ″passwords″ queue entity. If the steps above have been followed, the ″passwords″ entity should appear as a child of the ″PWStoreServer″ entity. Right click on the ″passwords″ node and select ″Properties″. On the note pane window that results, configure the following:
   - GENERAL

     **Enable Issue**
       Check the box.

     **Request PIN**
       The PIN used by the Tivoli Directory Integrator Storage Component Queue Manager to request certificate. This value must match the value for ″certServerReqPin″ in the `mqeconfig.props` file where the **mqeconfig** utility will execute.

## High Availability Considerations - Multiple Tivoli Directory Integrator MQe Password Connectors

For some deployments, it may be preferable to configure the Tivoli Directory Integrator MQe Password Connector, such that it supports a particular high availability requirement.

It is expected that an implementation supporting this type of requirement would employ multiple instances of the Tivoli Directory Integrator MQe Password Connector, each with its own associated MQe Queue Manager configuration. In this case multiple identical MQe server side configurations would be deployed allowing a network load balancer to route requests from the Tivoli Directory Integrator MQe Password Store client to an available server instance.

Each MQe Queue Manager on the server side will be configured using the **mqeconfig** utility. When this utility executes it will automatically request authentication certificates from the MQe Mini-Certificate server for the entities named ″PWStoreServer″ and ″PWStoreServer+passwords″. These represent the Queue Manager and Queue names respectively.

Before executing the second and each subsequent instance of the **mqeconfig** utility, it will necessary to re-enable certificate issue for the two MQe entities mentioned above. The steps for this are described below:
1. Open the MQe Mini-Certificate GUI.
2. Select the Tivoli Directory Integrator MQe Client Queue Manager entity. Right click on the ″PWStoreServer″ node and select ″Properties″. On the note pane window that results, configure the following:
   - GENERAL

**Enable Issue**
Check the box.

**Request Pin**
The PIN used by the Tivoli Directory Integrator Storage Component Queue Manager to request certificate. This value must match the value for ″certServerReqPin″ in the `mqeconfig.props` file where the **mqeconfig** utility will execute.

3. Select the ″passwords″ queue entity. If the steps above have been followed, the ″passwords″ entity should appear as a child of the ″PWStoreServer″ entity. Right click on the ″passwords″ node and select ″Properties″. On the note pane window that results, configure the following:
   - GENERAL

     **Enable Issue**
     Check the box.

     **Request Pin**
     The PIN used by the Tivoli Directory Integrator Storage Component Queue Manager to request certificate. This value must match the value for ″certServerReqPin″ in the `mqeconfig.props` file where the **mqeconfig** utility will execute.

## Certificate Renewal using the MQe Mini-Certificate Server

The certificates issued by the MQe Mini-Certificate server have a configurable validity period. The default validity period is 12 months. The MQe documentation states that issued certificates should be renewed before the period expires. To enable this, the **mqeconfig** utility includes an option to renew certificates for each entity that can be authenticated. For these entities, the entity names requiring certificate renewal are as follows:

**Tivoli Directory Integrator MQe Password Store Client side**
- ″PWStoreClient″ (Client side Queue Manager)
- ″PWStoreServer+passwords″ (Remote queue proxy on the client side)

**Tivoli Directory Integrator MQe Password Connector Server side**
- "PWStoreServer″ (Server side Queue Manager)
- "PWStoreServer+passwords″ (Real queue on the server side)

For each instance of Tivoli Directory Integrator MQe configurations, the following steps may be performed and repeated as required in order to renew previously issued authentication certificates. The MQe Mini-Certificate server that issued the original certificates must be running.

## Tivoli Directory Integrator Password Store Client Side

Performing the following steps on the Password Store Client Side renews the MQe Password Store certificates on the Client Side.

1. Open the MQe Mini-Certificate GUI.
2. Right click on the MQe authenticate-able entity for which certificate renewal is required, such as ″PWStoreServer+passwords″. On the note pane window that results, configure the following:
   - GENERAL

     **Enable Issue**
     Check the box.

**Request Pin**
: The PIN used by the Tivoli Directory Integrator Storage Component Queue Manager to request certificate. This value must match the value for ″certServerReqPin″ in the `mqeconfig.props` file where the **mqeconfig** utility will execute.

3. Use a text editor to set the following properties of the *`tdi install directory`*`/mqeconfig.props` files:

   **clientRootFolder**
   : The directory where MQe configuration instance is located.

   **certServerReqPin**
   : This value is used as a one time authentication PIN for the given authenticate-able entity when requesting certificate renewal from the MQe Mini-Certificate server. This value must match the ″Request PIN″ value from step 2 above.

   **certServerIPAndPort**
   : This value is used as the destination address for MQe Mini-Certificate server requests. The format of the value is ″`FastNetwork:`*`host`*`:`*`port`*″, where host must be the machine name or TCP IP address or host name where the MQe Mini-Certificate server is running.

   **certRenewalEntityName**
   : The MQe authenticate-able entity name (see above) requiring certificate renewal.

4. Open a command prompt in the *tdi install directory* of the Password Synchronizer, and enter the following command:

```
jvm\jre\bin\java -cp "./mqeconfig.jar" com.ibm.di.plugin.mqe.config.MQeConfig
     mqeconfig.props renewcert client
```

## Tivoli Directory Integrator Password Connector Server Side

Performing the following steps on the Password Store Server Side renews the MQe Password Store certificates on the Server Side.

1. Open the MQe Mini-Certificate GUI.
2. Right click on the MQe authenticate-able entity for which certificate renewal is required, such as ″PWStoreServer+passwords″. On the note pane window that results, configure the following:
   - GENERAL

     **Enable Issue**
     : Check the box.

     **Request Pin**
     : The PIN used by the Tivoli Directory Integrator Storage Component Queue Manager to request certificate. This value must match the value for ″certServerReqPin″ in the `mqeconfig.props` file where the **mqeconfig** utility will execute.
3. Use a text editor to set the following properties of the `<`*`tdi install directory`*`>/mqeconfig.props` files:

   **serverRootFolder**
   : The directory where MQe configuration instance is located.

   **certServerReqPin**
   : This value is used as a one time authentication PIN for the given authenticate-able entity when requesting certificate renewal from the MQe Mini-Certificate server. This value must match the ″Request PIN″ value from step 2 above.

**certServerIPAndPort**
: This value is used as the destination address for MQe Mini-Certificate server requests. The format of the value is ″`FastNetwork:`*`host`*`:`*`port`*″, where host must be the machine name or TCP IP address or host name where the MQe Mini-Certificate server is running.

**certRenewalEntityName**
: The MQe authenticate-able entity name (see above) requiring certificate renewal.

4. Open a command prompt in the *`tdi install directory`*`/plugins/jars` directory of the Password Connector, and enter the following command:

```
mqeconfig.{bat|sh} mqeconfig.props renewcert server
```

## Using the Password Store

The LDAP Password Store maintains state of the user's passwords. It keeps the passwords in the LDAP storage entries up to date with the passwords of the corresponding users. In contrast, the MQe Password Store does not maintain state of the user's passwords; it just reports the changes. Each message tells how the passwords of a user have changed, not what the user's password values are.

This difference is important for the design of the AssemblyLine that propagates the password changes to other systems, especially when multiple valued passwords are supported. In the case of the LDAP Password Store, the AssemblyLine must replace the passwords in the systems it keeps synchronized with the passwords read from the LDAP storage. When MQe Password Store is used, the AssemblyLine must duplicate just the reported password change on the other system.

Each MQe message contains the following information:

- User identifier (a string)
- Type of the password modification
- A list of password values

**User Identifier**
: The user identifier is the string value that identifies the user in the Target System (for LDAP Servers this is the LDAP distinguished name; for Windows this is the user account name). The AssemblyLine must locate the users on the systems that are synchronized based on this user identifier.

: **Note:** When the Target System is an LDAP Server, the MQe Password Store reports the whole LDAP distinguished name as user identifier (for example, ″cn=john,o=somecompany,c=us″), in contrast to the LDAP Password Store, where only the value of the first element (″john″) is used.

**Type of password modification and List of password values**
: The type of password modification might be one of **replace**, **add** or **delete** and correspondingly indicates that the password values have been replaced, that new values have been added, or certain values have been deleted.

: **add** and **delete** make sense only when multiple password values are supported by the Target System. If the Target System does not support multiple passwords for a single user, the type is always **replace**.

Depending on the type of password modification, the list of password values means the following:

**replace**
The passwords for the specified user are replaced with the passwords specified in the list of password values

**add** The passwords from the specified list of password values are added to the user's passwords (for example, new passwords are created for this user and the old ones are still in effect)

**delete** The passwords from the specified list of passwords values are removed from the user's passwords (for example, some of the user's passwords are deleted and the user can no longer use them)

**Note:** The type of password modification refers to the password attribute, not to the entry or user for which the password is modified. Thus **add** means that new password values are added to the user's password attribute and not that a new user is added in the system. On the other hand, when a new user is added in the system, it is appropriate to receive modification type **replace** because of the way the user password is internally set in the Target System.

## Availability issues

The QueueManager on the Storage Component is automatically started and stopped when the Storage Component is started and stopped.

The QueueManager on the IBM Tivoli Directory Integrator is automatically started and stopped when the MQe Password Store Connector is correspondingly initialized and stopped. This means that the QueueManager on the Storage Component is available only when the Storage Component is available and the QueueManager on the IBM Tivoli Directory Integrator is available only when the AssemblyLine with the MQe Password Store Connector is running.

There are three interesting cases regarding solution components availability:

**Both QueueManagers are available (the Password Synchronizer is running and the AssemblyLine is running)**
Each new intercepted password is immediately transferred between the QueueManagers and retrieved by the MQe Password Store Connector.

**Only the QueueManager on the Storage Component is available (the Password Synchronizer is running and the AssemblyLine is not running)**
Each new intercepted message is stored on the local disk by the Storage Component QueueManager. When the AssemblyLine is started, all messages stored offline are automatically transferred to the QueueManager on the IBM Tivoli Directory Integrator and the MQe Password Store Connector retrieves them from there.

**Only the QueueManager on the IBM Tivoli Directory Integrator is available (the Password Synchronizer is not running and the AssemblyLine is running)**
There are no new messages in this case because the Password Synchronizer is not running. When the Password Synchronizer is started, all messages stored previously on the Storage Component QueueManager are automatically transferred to the QueueManager on the IBM Tivoli Directory Integrator and the MQe Password Store Connector retrieves them from there.

No messages (password updates) are lost regardless of the availability of the Password Synchronizer and the MQe Password Store Connector and when they are started and stopped. However, for message transfer to take place, both QueueManagers must be available at the same time for at least a few minutes.

# Chapter 10. Tivoli Identity Manager Integration

This chapter describes the configuration of the Tivoli Identity Manager Integration for the Sun ONE Directory Server Password Synchronizer, IBM Directory Server Password Synchronizer, Domino HTTP Password Synchronizer and Password Synchronizer for UNIX and Linux.

This chapter contains the following sections:
- "Overview"
- "Configuring Password Synchronizers for Tivoli Identity Manager Integration"
- "Known Issues" on page 113

## Overview

The Tivoli Identity Manager Integration for the Password Sychronizers allows synchronized passwords to be verified by a Tivoli Identity Manager Server's Password Strength Servlet prior to synchronization. This allows Password Synchronization to incorporate password complexity checking via Tivoli Identity Manager Password Policies.

The Tivoli Identity Manager Integration is enabled by utilizing one of two of the Tivoli Identity Manager Decorator Password Synchronizer classes:
- LDAPPasswordSynchronizerITIMDecorator
- MQePasswordStoreITIMDecorator

### Supported Synchronizers

The Tivoli Identity Manager Password Synchronizer Decorator classes are supported by the following Password Synchronizers:
- Sun ONE Directory Server Password Synchronizer,
- IBM Directory Server Password Synchronizer,
- Domino HTTP Password Synchronizer,
- PAM Password Synchronizer.

## Configuring Password Synchronizers for Tivoli Identity Manager Integration

Configure the Password Synchronizer to utilize a Tivoli Identity Manager Decorator by setting the **syncClassName** property value within the Password Synchronizer Configuration file to one of the Decorator classnames shown below:
- com.ibm.di.plugin.pwsync.LDAPPasswordSynchronizerITIMDecorator
- com.ibm.di.plugin.mqe.store.MQePasswordStoreITIMDecorator

Depending on the Password Store used by the Password Synchronizer, the Tivoli Identity Manager Decorator will either be configured in the MQe Password Store configuration file (`mqeconfig.props`) or LDAP Password Store configuration file (`idipwsync.props`) for the **MQePasswordStoreITIMDecorator** and **LDAPPasswordSynchronizerITIMDecorator** Decorators respectively. The configuration files contain properties in the format *property_name=property_value*, with each property placed on a single line.

Specify the following properties (property names are case sensitive):

**itimPasswordUrl**
URL of the Tivoli Identity Manager hosted Password Strength Servlet. For example:

```
https://host/passwordsync/sync
```

Needed if a decorator classname is specified for **syncClassName** in the Password Synchronizer Configuration file.

**itimPrincipalName**
Tivoli Identity Manager user name permitted to perform a password check. Needed if a decorator classname is specified for **syncClassName** in the Password Synchronizer Configuration file.

**itimPrincipalPassword**
The password for the Tivoli Identity Manager user name within **itimPrincipalName**. Needed if a decorator classname is specified for **syncClassName** in `pampwsync.props`.

**itimSourceDN**
The Tivoli Identity Manager service name against which the password check should be performed. For example:

```
erservicename=TDIPasswordService, o=IBM, ou=IBM, dc=com
```

Needed if a decorator classname is specified for **syncClassName** in the Password Synchronizer Configuration file.

**itimKeyDatabase**
The location of the key database file containing the signer certificate for the Tivoli Identity Manager server certificate. Needed if a decorator classname is specified for **syncClassName** in the Password Synchronizer Configuration file.

**Note:** When the LDAP Tivoli Identity Manager Integration is active, any LDAP SSL and Tivoli Identity Manager SSL Certificates must reside in the same keystore. See the example LDAP Password Store configuration file below.

**itimKeyDatabasePassword**
The password for the **itimKeyDatabase**. Needed if a decorator classname is specified for **syncClassName** in the Password Synchronizer Configuration file.

The following is a sample MQe Password Store configuration file configured to use Tivoli Identity Manager Integration:

```
#MQe PasswordStore Properties (with encoded passwords)
#Wed Jul 27 15:43:51 EST 2005

debug=true
logFile=/opt/ibm/DiPlugins/PAM/pam_mqe.log
notificationPort=41002
qmIniFileName=/opt/ibm/DiPlugins/PAM/MQePWStore/pwstore_client.ini

encrypt=true
encryptKeyStoreCertificate=idicryptotest
encryptKeyPassword120be1f2046d0633
encryptKeyStoreFilePassword=120be1f2046d0633
encryptKeyStoreFilePath=/opt/ibm/DiPlugins/PAM/idicrypto.jks

itimPrincipalName=ITIM Manager
```

```
itimPrincipalPassword=120be1f2046d0633
itimSourceDN=erservicename= TDIPasswordService,o=International
   Business Machines,ou=IBM,dc=com
itimPasswordUrl=https://itim_host:9443/passwordsynch/synch
itimSslKeyDatabase=/opt/ibm/DiPlugins/PAM/idicrypto.jks
itimSslKeyDatabasePassword=120be1f2046d0633
```

The following is a sample LDAP Password Store configuration file configured to use Tivoli Identity Manager Integration:

```
#IBM Directory Integrator LDAP Password Store Settings with Encoded Passwords
#Wed Jul 27 15:43:51 EST 2005

host=ldap_host
port=636
ldapLogInUserId=cn=root
ldapLogInPassword=120be1f2046d0633

suffix=o=ibm,c=au
waitForStore=true

ssl=true
sslKeyStoreFilePassword=120be1f2046d0633
sslKeyStoreFilePath=/opt/ibm/DiPlugins/PAM/idicrypto.jks

encrypt=true
encryptKeyStoreCertificate=idicryptotest
encryptKeyPassword=120be1f2046d0633
encryptKeyStoreFilePassword=120be1f2046d0633
encryptKeyStoreFilePath=/opt/ibm/DiPlugins/PAM/idicrypto.jks

itimPrincipalName=ITIM Manager
itimPrincipalPassword=120be1f2046d0633
itimSourceDN=erservicename=TDIPasswordService,o=International
   Business Machines,ou=IBM,dc=com
itimPasswordUrl=https://itim_host:9443/passwordsynch/synch
itimSslKeyDatabase=/opt/ibm/DiPlugins/PAM/idicrypto.jks
itimSslKeyDatabasePassword=120be1f2046d0633

logFilePath=/opt/ibm/DiPlugins/PAM/idipwsync.log
```

**Note:** *itim_host* and *ldap_host* are variables. Do not copy and paste this example without first replacing variables with accurate data.

## Known Issues

Be aware of the following issues which are known to exist at time of publication:

- When Tivoli Identity Manager Integration is enabled **checkRepository** must be set to *true* in the Password Synchronizer Configuration file.
- When the LDAP Tivoli Identity Manager Integration is active, any LDAP SSL and Tivoli Identity Manager SSL Certificates must reside in the same keystore.
- The Tivoli Identity Manager Password Synchronizer Decorators are not supported on Domino 6.5.4.
- To utilize the Tivoli Identity Manager Password Synchronizer on Domino 7.0 the Password Synchronizer must be started independently using a command similar to the one shown below:

```
/opt/ibm/lotus/notes/latest/ibmpow/jvm/bin/java -cp /local/notesdata/
   idipwsync -DsyncClass=com.ibm.di.plugin.mqe.store.
   MQePasswordStoreITIMDecorator -DserverPort=19011 -DjavaLogFile=/
   local/notesdata/idipwsync/proxy.log -Ddebug=true
   com.ibm.di.plugin.pwsync.Proxy
```

# Appendix A. AIX 5.2 PAM Configuration

AIX 5.2 does not natively support the Pluggable Authentication Module (PAM) Framework; instead PAM is supported via the Loadable Authentication Module (LAM) Framework.

The following steps outline how to enable PAM via the LAM Framework on AIX 5.2 to function with the PAM Password Synchronizer:

1. Copy the PAM Module to the system.
2. Place the 32-bit version of the PAM Module into the `/usr/lib/security` directory.
3. Place the 64-bit version of the PAM Module into the `/usr/lib/security/64` directory.
4. Ensure that the 32-bit and 64-bit Module share the same file name. For example:
   - `/usr/lib/security/libpamtivoli.so`
   - `/usr/lib/security/64/libpamtivoli.so`

   **Note:** Only use the filename of the modules (both should have the same filenames), not a fully qualified path to the module for the **module_path** section of the Password Synchronizer PAM Module entry. When the PAM Framework loads your module it will either use the 32-bit or 64-bit version, depending on the Operating System mode. If the Operating System is in 32-bit mode then the path to the Password Synchronizer PAM module (`pamlibtivoli.so`) will be resolved to `/usr/lib/security/pamlibtivoli.so`. However if the Operating System is in 64-bit mode, the path to the module will be resolved to `/usr/lib/security/64/pamlibtivoli.so`. Specifying an absolute path to the Password Synchronizer PAM Module binds the PAM Framework to one specific mode only (32-bit or 64-bit).
5. Set the permissions and ownership of the 32-bit and 64-bit modules as follows:
   ```
   chmod 555 /usr/lib/security/libpamtivoli.so
   chown root:system /usr/lib/security/libpamtivoli.so
   chmod 555 /usr/lib/security/64/libpamtivoli.so
   chown root:system /usr/lib/security/64/libpamtivoli.so
   ```
6. Create an empty PAM Framework configuration file (`/etc/pam.conf`). Set the permissions and ownership of the configuration file:
   ```
   touch /etc/pam.conf
   chmod 644 /etc/pam.conf
   chown root:security /etc/pam.conf
   ```
7. Update the PAM Framework configuration file (`/etc/pam.conf`) to include a basic Module Stack (including the PAM Password Synchronizer Module):
   ```
   #
   # Authentication
   #
   ftp     auth    required        /usr/lib/security/pam_aix
   imap    auth    required        /usr/lib/security/pam_aix
   login   auth    required        /usr/lib/security/pam_aix
   rexec   auth    required        /usr/lib/security/pam_aix
   rlogin  auth    required        /usr/lib/security/pam_aix
   snapp   auth    required        /usr/lib/security/pam_aix
   su      auth    required        /usr/lib/security/pam_aix
   telnet  auth    required        /usr/lib/security/pam_aix
   ```

```
OTHER   auth    required        /usr/lib/security/pam_aix

#
# Account Management
#
ftp     account required        /usr/lib/security/pam_aix
login   account required        /usr/lib/security/pam_aix
rexec   account required        /usr/lib/security/pam_aix
rlogin  account required        /usr/lib/security/pam_aix
rsh     account required        /usr/lib/security/pam_aix
su      account required        /usr/lib/security/pam_aix
telnet  account required        /usr/lib/security/pam_aix
OTHER   account required        /usr/lib/security/pam_aix

#
# Password Management
#
login   password  required      /usr/lib/security/pam_aix
passwd  password  required      libpamtivoli.so use_first_pass
     /opt/ibm/DiPlugins/PAM/pampwsync.props
rlogin  password  required      /usr/lib/security/pam_aix
su      password  required      /usr/lib/security/pam_aix
telnet  password  required      /usr/lib/security/pam_aix
OTHER   password  required      /usr/lib/security/pam_aix

#
# Session Management
#
ftp     session required        /usr/lib/security/pam_aix
imap    session required        /usr/lib/security/pam_aix
login   session required        /usr/lib/security/pam_aix
rexec   session required        /usr/lib/security/pam_aix
rlogin  session required        /usr/lib/security/pam_aix
rsh     session required        /usr/lib/security/pam_aix
snapp   session required        /usr/lib/security/pam_aix
su      session required        /usr/lib/security/pam_aix
telnet  session required        /usr/lib/security/pam_aix
OTHER   session required        /usr/lib/security/pam_aix
```

8. Enable the PAM Framework:
   a. Add the following lines to the end of the `/usr/lib/security/methods.cfg` file:

      **PAM:**
      ```
      program = /usr/lib/security/PAM
      program_64 = /usr/lib/security/PAM
      ```

      **PAMfiles:**
      ```
      options = auth=PAM,db=BUILTIN
      ```
   b. Test the PAM Framework has been successfully enabled by changing a user's password, if PAM has been successfully configured executing the commands below should cause the user's password to be synchronized to the Password Store.
      ```
      useradd testuser
      passwd testuser
      telnet localhost
      ```

# Appendix B. AIX 5.3 PAM Configuration

AIX 5.3 natively supports the Pluggable Authentication Module (PAM) Framework; Post-installation AIX is configured to utilize Standard Authentication rather than PAM.

The following steps outline how to enable PAM on AIX 5.3:

1. To enable PAM open the `/etc/security/login.cfg` file and change the value of the **auth_type** property within the **usw** stanza to *PAM_AUTH*. For example:
   ```
   usw:
     shells = ...
     maxlogins = 32767
     logintimeout = 60
     auth_type = PAM_AUTH
   ```
2. Copy the PAM Module to the system.
3. Place the 32-bit version of the PAM Module into the `/usr/lib/security` directory.
4. Place the 64-bit version of the PAM Module into the `/usr/lib/security/64` directory.
5. Ensure that the 32-bit and 64-bit Module share the same file name. For example:
   - `/usr/lib/security/libpamtivoli.so`
   - `/usr/lib/security/64/libpamtivoli.so`

   **Note:** Only use the filename of the modules (both should have the same filenames), not a fully quantified path to the module for the **module_path** section of the Password Synchronizer PAM Module entry. When the PAM Framework loads your module it will either use the 32-bit or 64-bit version, depending on the Operating System mode. If the Operating System is in 32-bit mode then the path to the Password Synchronizer PAM module (`pamlibtivoli.so`) will be resolved to `/usr/lib/security/pamlibtivoli.so`. However if the Operating System is in 64-bit mode, the path to the module will be resolved to `/usr/lib/security/64/pamlibtivoli.so`. Specifying an absolute path to the Password Synchronizer PAM Module binds the PAM Framework to one specific mode only (32-bit or 64-bit).
6. Set the permissions and ownership of the 32-bit and 64-bit modules as follows:
   ```
   chmod 555 /usr/lib/security/libpamtivoli.so
   chown root:system /usr/lib/security/libpamtivoli.so
   chmod 555 /usr/lib/security/64/libpamtivoli.so
   chown root:system /usr/lib/security/64/libpamtivoli.so
   ```
7. Update the `/etc/pam.conf` file to include the PAM Password Synchronizer Module in the Password Management stack. The following excerpt from the modified AIX 5.3 PAM Framework configuration file shows how your configuration should look:
   ```
   login password  required /usr/lib/security/pam_aix
   passwd password  required libpamtivoli.so use_first_pass
       /opt/ibm/DiPlugins/PAM/pampwsync.props
   rlogin password  required /usr/lib/security/pam_aix
   su password  required /usr/lib/security/pam_aix
   telnet password  required /usr/lib/security/pam_aix
   OTHER password  required /usr/lib/security/pam_prohibit
   ```

8. Test the PAM Framework has been successfully enabled by changing a user’s password, if PAM has been successfully configured executing the commands below should cause the user’s password to be synchronized to the Password Store.

   ```
   useradd testuser
   passwd testuser
   telnet localhost
   ```

# Appendix C. IBM Software Support

IBM Software Support provides assistance with product defects.

Before contacting IBM Software Support, your company must have an active IBM software maintenance contract, and you must be authorized to submit problems to IBM. The type of software maintenance contract that you need depends on the type of product you have:

- For IBM distributed software products (including, but not limited to, Tivoli, Lotus, and Rational® products, as well as DB2® and WebSphere products that run on Windows or UNIX operating systems), enroll in Passport Advantage® in one of the following ways:
  – **Online**: Go to the following Passport Advantage Web page and click **How to Enroll**:

    http://www.lotus.com/services/passport.nsf/WebDocs/Passport_Advantage_Home
  – **By phone**: For the phone number to call in your country, go to the IBM Software Support Web site (http://techsupport.services.ibm.com/guides/contacts.html) and click the name of your geographic region.
- For IBM eServer™ software products (including, but not limited to, DB2 and WebSphere products that run in zSeries®, pSeries®, and iSeries™ environments), you can purchase a software maintenance agreement by working directly with an IBM sales representative or an IBM Business Partner. For more information about support for eServer software products, go to the IBM Technical Support Advantage Web page (http://www.ibm.com/servers/eserver/techsupport.html).

If you are not sure what type of software maintenance contract you need, call 1-800-IBMSERV (1-800-426-7378) in the United States or, from other countries, go to the contacts page of the IBM Software Support Handbook on the Web (http://techsupport.services.ibm.com/guides/contacts.html) and click the name of your geographic region for phone numbers of people who provide support for your location.

Follow the steps in this topic to contact IBM Software Support:

1. "Determine the business impact of your problem"
2. "Describe your problem and gather background information" on page 120
3. "Submit your problem to IBM Software Support" on page 120

## Determine the business impact of your problem

When you report a problem to IBM, you are asked to supply a severity level. Therefore, you need to understand and assess the business impact of the problem you are reporting. Use the following criteria:

| | |
|---|---|
| **Severity 1** | **Critical** business impact: You are unable to use the program, resulting in a critical impact on operations. This condition requires an immediate solution. |
| **Severity 2** | **Significant** business impact: The program is usable but is severely limited. |

| | |
|---|---|
| **Severity 3** | **Some** business impact: The program is usable with less significant features (not critical to operations) unavailable. |
| **Severity 4** | **Minimal** business impact: The problem causes little impact on operations, or a reasonable circumvention to the problem has been implemented. |

## Describe your problem and gather background information

When explaining a problem to IBM, be as specific as possible. Include all relevant background information so that IBM Software Support specialists can help you solve the problem efficiently. To save time, know the answers to these questions:

- What software versions were you running when the problem occurred?
- Do you have logs, traces, and messages that are related to the problem symptoms? IBM Software Support is likely to ask for this information.
- Can the problem be recreated? If so, what steps led to the failure?
- Have any changes been made to the system? (For example, hardware, operating system, networking software, and so on.)
- Are you currently using a workaround for this problem? If so, please be prepared to explain it when you report the problem.

## Submit your problem to IBM Software Support

You can submit your problem in one of two ways:

- **Online**: Go to the ″Submit and track problems″ page on the IBM Software Support site (http://www.ibm.com/software/support/probsub.html). Enter your information into the appropriate problem submission tool.
- **By phone**: For the phone number to call in your country, go to the contacts page of the IBM Software Support Handbook on the Web (http://techsupport.services.ibm.com/guides/contacts.html) and click the name of your geographic region.

If the problem you submit is for a software defect or for missing or inaccurate documentation, IBM Software Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Software Support provides a workaround for you to implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the IBM product support Web pages daily, so that other users who experience the same problem can benefit from the same resolutions.

For more information about problem resolution, see “Searching knowledge bases” and “Obtaining fixes” on page 121.

## Searching knowledge bases

If you have a problem with your IBM software, you want it resolved quickly. Begin by searching the available knowledge bases to determine whether the resolution to your problem is already documented.

### Search the information center on your local system or network

IBM provides extensive documentation that can be installed on your local machine or on an intranet server. You can use the search function of this information center to query conceptual information, instructions for completing tasks, reference information, and support documents.

### Search the Internet

If you cannot find an answer to your question in the information center, search the Internet for the latest, most complete information that might help you resolve your problem. To search multiple Internet resources for your product, expand the product folder in the navigation frame to the left and select **Support on the Web**. From this topic, you can search a variety of resources including:

- IBM technotes
- IBM downloads
- IBM Redbooks™
- IBM DeveloperWorks
- Forums and newsgroups
- Google

## Obtaining fixes

A product fix might be available to resolve your problem. You can determine what fixes are available for your IBM software product by checking the product support Web site:

1. Go to the IBM Software Support Web site (http://www.ibm.com/software/support).
2. Under **Products A - Z**, select your product name. This opens a product-specific support site.
3. Under **Self help**, follow the link to **All Updates**, where you will find a list of fixes, fix packs, and other service updates for your product. For tips on refining your search, click **Search tips**.
4. Click the name of a fix to read the description and optionally download the fix.

To receive weekly e-mail notifications about fixes and other news about IBM products, follow these steps:

1. From the support page for any IBM product, click **My support** in the upper-right corner of the page.
2. If you have already registered, skip to the next step. If you have not registered, click register in the upper-right corner of the support page to establish your user ID and password.
3. Sign in to **My support**.
4. On the My support page, click **Edit profiles** in the left navigation pane, and scroll to **Select Mail Preferences**. Select a product family and check the appropriate boxes for the type of information you want.
5. Click **Submit**.
6. For e-mail notification for other products, repeat Steps 4 and 5.

For more information about types of fixes, see the *Software Support Handbook* (http://techsupport.services.ibm.com/guides/handbook.html).

# Appendix D. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## Trademarks