Configuration Guide

Version 4.5.1
Note:
Before using this information and the product it supports, read the information in "Notices," on page 165.
## Contents

**Preface** ................................................................. vii
- Who Should Read This Book ........................................... vii
- Publications ................................................................. vii
  - Tivoli Identity Manager Library ................................... vii
  - Prerequisite Product Publications ................................. viii
  - Related Publications .................................................. viii
  - Accessing Publications Online ...................................... ix
- Accessibility ............................................................... x
- Contacting Software Support ......................................... x
- Conventions Used in This Book ....................................... x
  - Typeface Conventions .................................................. x
  - Operating System Differences ....................................... x
  - Revision Bars used in the Version 4.5.1 Library ............... xi
  - Definitions for HOME Directory Variables ....................... xi

### Part 1. Basic System Configuration .................................... 1

**Chapter 1. Using the System Configuration Tool (runConfig)** .......... 3
- System Configuration Tool User Interface ................................ 4
  - Property Files .......................................................... 5
  - Starting the System Configuration Tool (runConfig) .................. 5
- General System Properties .............................................. 6
  - Application Server ...................................................... 6
  - Scheduling Information ............................................... 7
- Directory Properties ................................................... 7
  - Tivoli Identity Manager Server Directory Connection Information ... 8
  - LDAP Connection Pool Information .................................. 8
- Database Properties ................................................... 8
  - General Database Information ......................................... 9
  - Database Pool Information ........................................... 9
- Logging Properties .................................................... 10
  - Logging Level ........................................................... 10
  - Trace Exceptions ...................................................... 10
- Mail Properties ........................................................ 10
  - Web Server Information ................................................. 11
  - Mail Information ........................................................ 12
  - Mail Server Information .............................................. 12
- User Interface Properties ............................................. 12
  - Customer Logo and Customer Logo Link ............................ 12
  - List Page Size .......................................................... 12
- Security Properties .................................................... 13
  - Encryption Settings .................................................... 13
  - Application Server User Management Settings ..................... 13
  - Configuring Properties from the Tivoli Identity Manager GUI ... 14

**Chapter 2. Customizing the Tivoli Identity Manager GUI** ............. 15
- Configuring a Custom Logo ............................................. 15
  - Adding a Logo to the Tivoli Identity Manager GUI ............... 15
- Customizing List Displays ............................................. 15
  - Setting the Maximum Number of Items per Page ................... 16
  - Setting the Maximum Number of Links per Page .................... 16
- Custom Display Attributes ............................................ 16
  - Customizing Fonts and Colors ....................................... 16
Chapter 7. Configuring Crystal Reports. .................................................. 67
Process Flow When Using Crystal Reports ........................................... 68
Configuring Tivoli Identity Manager to Use Crystal Reports .................... 69
  1. RAS Configuration (supported on Windows platform only) .......... 69
  2a. Tivoli Identity Manager (WebSphere on Windows) Configuration ...... 69
  2b. Tivoli Identity Manager (WebSphere on UNIX) Configuration ....... 71
  2c. Tivoli Identity Manager Server (WebLogic on Windows) Configuration 73
  2d. Tivoli Identity Manager (WebLogic on UNIX) Configuration ....... 74
  3. Client Configuration (supported on Windows platform only) .......... 75

Chapter 8. Designing Crystal Report Filter Conditions .......................... 77
Tutorial on Designing Filter Conditions ................................................. 77
Example Reports ................................................................................. 79
  Specifying JOIN conditions in Report Designer while designing reports . 79

Chapter 9. Designing Custom Report Filter Conditions .......................... 83
Tutorial on Designing Filter Conditions ................................................. 83
Example Reports ................................................................................. 85
  Use of functions in reports ............................................................... 85
  Specifying JOIN conditions in Report Designer while designing reports . 85

Part 4. Property File Reference .............................................................. 89

Chapter 10. Configuring System Properties ............................................. 91
Understanding Properties Files ............................................................... 92
WebLogic-specific Configuration ........................................................... 93
WebSphere-specific Configuration .......................................................... 97
Application Server Information ............................................................. 101
Default Tenant Information ................................................................. 103
LDAP Server Information ................................................................. 104
Cache Information .............................................................................. 105
Messaging Information ................................................................. 106
Scheduling Information ............................................................... 109
Password Transaction Monitor Settings .............................................. 110
XML and DTD Information ............................................................... 111
LDAP Connection Pool Information .................................................. 112
Encryption Information ................................................................. 113
System Configuration Program ...................................................... 114
Workflow Configuration Information ................................................. 115
Mail Services Configuration ............................................................. 119
Reconciliation Information ............................................................... 120
Shared Secret Hashing ................................................................. 122
SSL Two-way Authentication Properties ....................................... 123
Request Management UI Configuration ............................................. 124
  Application Client Request Configuration ......................................... 125

Chapter 11. Configuring Supplemental Properties .................................... 127
Understanding Properties Files ............................................................. 128
  adhocreporting.properties ............................................................. 129
  crystal.properties ............................................................................. 132
  DataBaseFunctions.conf ............................................................... 133
enRoleAuthentication.properties ........................................ 134
  Configuring a custom authentication mechanism ....................... 135
enRoleDatabase.properties .............................................. 137
enRoleLDAPConnection.properties ..................................... 141
enRoleLogging.properties ................................................ 144
enRoleMail.properties ................................................... 147
enrolepolicies.properties ................................................ 149
enroleworkflow.properties ............................................... 151
fesiextensions.properties .............................................. 153
UI.properties ................................................................ 155
CustomLabels.properties .................................................. 158
Additional property files .................................................. 158

Part 5. Appendixes ................................................................. 163

Appendix. Notices ............................................................... 165
  Trademarks .................................................................. 166

Glossary ........................................................................... 169

Index ............................................................................... 173
Preface

Welcome to the IBM® Tivoli® Identity Manager Configuration Guide.

Who Should Read This Book

This manual is intended for system and security administrators who install, maintain, or administer software on their site’s computer systems. Readers are expected to understand system and security administration concepts. Additionally, the reader should understand administration concepts for the following:

- Directory server
- Database server
- WebSphere® embedded messaging support
- WebSphere Application Server or WebLogic
- IBM HTTP Servers

Publications

Read the descriptions of the Tivoli Identity Manager library, the prerequisite publications, 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.

Tivoli Identity Manager Library

The publications in the Tivoli Identity Manager technical documentation library are organized into the following categories:

- Release Information
- Online User Assistance
- Server Installation
- Administration and Configuration
- Technical Supplements
- Agent Installation

Release Information:

- IBM Tivoli Identity Manager Release Notes
  Provides software and hardware requirements for Tivoli Identity Manager, and additional fix, patch, and other support information.
- Tivoli Identity Manager Read This First Card

Online User Assistance:

- Online user assistance for Tivoli Identity Manager
  Provides integrated online help topics for all Tivoli Identity Manager administrative tasks.

Server Installation:

- IBM Tivoli Identity Manager Server Installation Guide on UNIX and Linux using WebSphere
  Provides installation information for Tivoli Identity Manager.
• *IBM Tivoli Identity Manager Server Installation Guide on Windows using WebSphere*
  Provides installation information for Tivoli Identity Manager.

• *IBM Tivoli Identity Manager Server Installation Guide on UNIX using WebLogic*
  Provides installation information for Tivoli Identity Manager.

• *IBM Tivoli Identity Manager Server Installation Guide on Windows 2000 using WebLogic*
  Provides installation information for Tivoli Identity Manager.

**Administration and Configuration:**

• *IBM Tivoli Identity Manager Policy and Organization Administration Guide*
  Provides topics for Tivoli Identity Manager administrative tasks.

• *IBM Tivoli Identity Manager End User Guide*
  Provides beginning user information for Tivoli Identity Manager.

• *IBM Tivoli Identity Manager Configuration Guide*
  Provides configuration information for single-server and cluster Tivoli Identity Manager configurations.

**Technical Supplements:**

• *IBM Tivoli Identity Manager Problem Determination Guide*
  Provides additional problem solving information for the Tivoli Identity Manager product.

**Agent Installation:**

• The Tivoli Identity Manager technical documentation library also includes an evolving set of platform-specific installation documents for the Agent component of a Tivoli Identity Manager implementation.

**Prerequisite Product Publications**

To use the information in this book effectively, you must have knowledge of the products that are prerequisites for Tivoli Identity Manager. Publications are available from the following locations:

• WebSphere Application Server

  **Note:** The following brief list of Redbooks describes installing and configuring WebSphere Application Server and providing additional security. Although the list was current when this publication went to production, publications may become obsolete. Contact your customer representative for a recommended list of resource information.
  – *IBM WebSphere Application Server V5.0 System Management and Configuration*, an IBM Redbook
  – *IBM WebSphere Application Server V5.0 Security*, an IBM Redbook

• WebLogic Server
  [http://e-docs.bea.com/](http://e-docs.bea.com/)

• Database servers
  – IBM DB2
    [http://www.ibm.com/software/data/db2](http://www.ibm.com/software/data/db2)
  – Oracle
Related Publications

Information related to Tivoli Identity Manager Server is available in the following publications:

- 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:
  

- The Tivoli Software Glossary includes definitions for many of the technical terms related to Tivoli software. The Tivoli Software Glossary is available, in English only, from the Glossary link on the left side of the Tivoli Software Library Web page at:
  

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:


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.

Product publications include release notes, installation guides, user’s guides, administrator’s guides, and developer’s references.

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

The product documentation includes the following features to aid accessibility:

- Documentation is available in both HTML and convertible PDF formats to give the maximum opportunity for users to apply screen-reader software.
- All images in the documentation are provided with alternative text so that users with vision impairments can understand the contents of the images.

Contacting Software Support

Before contacting IBM Tivoli Software Support with a problem, refer to the IBM Tivoli Software Support site by clicking the Tivoli support link at the following Web site:

http://www.ibm.com/software/support/

If you need additional help, contact 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

Conventions Used in this Book

This reference uses several conventions for special terms and actions and for operating system-dependent commands and paths.

Typeface Conventions

The following typeface conventions are used in this reference:

**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.

Operating System Differences

This book uses the UNIX convention for specifying environment variables and for directory notation. When using the Windows command line, replace $variable with %variable% for environment variables and replace each forward slash (/) with a backslash (\) in directory paths. If you are using the bash shell on a Windows system, you can use the UNIX conventions.
Revision Bars used in the Version 4.5.1 Library

The Tivoli Identity Manager version 4.5.1 technical documentation library makes use of revision bar characters to indicate where technical changes have occurred to the information previously found in the version 4.5 library. Revision bars are indicated by a vertical line ( | ) in the page margin to the left of the change.

Definitions for HOME Directory Variables

The following table contains the default definitions used in this document to represent the "HOME" directory level for various product installation paths. You can customize the installation directory and HOME directory for your specific implementation. If this is the case, you need to make the appropriate substitution for the definition of each variable represented in this table.

<table>
<thead>
<tr>
<th>Path Variable</th>
<th>Default Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ITIM_HOME</strong></td>
<td>Windows: c:\itim45\ UNIX: /itim45/</td>
</tr>
<tr>
<td></td>
<td><strong>WAS_HOME</strong></td>
</tr>
<tr>
<td></td>
<td><strong>WAS_NDM_HOME</strong></td>
</tr>
<tr>
<td></td>
<td><strong>BEA_HOME</strong></td>
</tr>
</tbody>
</table>
## Part 1. Basic System Configuration

### Chapter 1. Using the System Configuration Tool (runConfig)
- System Configuration Tool User Interface ............................................. 3
  - Property Files ................................................................. 5
- Starting the System Configuration Tool (runConfig) ......................... 5
  - Starting the System Configuration Tool (UNIX) .......................... 5
- General System Properties ....................................................... 6
  - Application Server ......................................................... 6
  - Scheduling Information .................................................... 7
    - Heart Beat ..................................................................... 7
  - Recycle Bin Age Limit ....................................................... 7
- Directory Properties ................................................................. 7
  - Tivoli Identity Manager Server Directory .................................. 8
  - Connection Information ...................................................... 8
    - LDAP Connection Pool Information .................................... 8
- Database Properties ................................................................. 8
  - General Database Information .............................................. 9
    - Database Pool Information ............................................. 9
- Logging Properties ................................................................. 10
  - Logging Level .................................................................... 10
    - Trace Exceptions .......................................................... 10
- Mail Properties ................................................................. 10
  - Web Server Information ..................................................... 11
  - Mail Information .............................................................. 12
  - Mail Server Information .................................................... 12
- User Interface Properties ....................................................... 12
  - Customer Logo and Customer Logo Link ................................ 12
  - List Page Size ................................................................. 12
- Security Properties ............................................................... 13
  - Encryption Settings ......................................................... 13
    - Application Server User Management Settings ..................... 13
- Configuring Properties from the Tivoli Identity Manager GUI ............. 14

### Chapter 2. Customizing the Tivoli Identity Manager GUI
- Configuring a Custom Logo ..................................................... 15
  - Adding a Logo to the Tivoli Identity Manager GUI ......... 15
- Customizing List Displays ....................................................... 15
  - Setting the Maximum Number of Items per Page .................. 16
    - Setting the Maximum Number of Links per Page .............. 16
- Custom Display Attributes ..................................................... 16
  - Customizing Fonts and Colors ........................................... 16

### Chapter 3. Configuring Account and Password Management
- Configuring Suspend/Restore Person with Accounts .................... 17
- Configuring Optional Password on Person/Account ..................... 17
- Restore .................................................................................. 17
  - Specifying Invalid Passwords .......................................... 18
    - Adding Words to the Password Dictionary .................... 18
  - Excluding Accounts from Reconciliation ............................ 18

Selecting Accounts to Exclude from Reconciliations .................... 19
E-mail Template Customization .................................................. 19

© Copyright IBM Corp. 2004
Chapter 1. Using the System Configuration Tool (runConfig)

The Tivoli Identity Manager Server is configured through the management of various system properties. Each system property represents a value that is used to control how the Tivoli Identity Manager Server behaves. For example, a system property can be used to determine whether a correct challenge response immediately allows a user to log in or whether it should send an e-mail with a new password.

System properties are configured in the following ways:
• Use the system configuration utility, runConfig (described in this chapter)
• Manually modify properties by editing the appropriate property file:
  – For detailed information about system properties (enrole.properties), see Chapter 10, “Configuring System Properties,” on page 91
  – For detailed information about supplemental properties, see Chapter 11, “Configuring Supplemental Properties,” on page 127

Section topics:
• “System Configuration Tool User Interface” on page 4
• “General System Properties” on page 6
• “Directory Properties” on page 7
• “Database Properties” on page 8
• “Logging Properties” on page 10
• “Mail Properties” on page 10
• “User Interface Properties” on page 12
• “Security Properties” on page 13
• “Configuring Properties from the Tivoli Identity Manager GUI” on page 14
System Configuration Tool User Interface

This section provides information about the Tivoli Identity Manager system configuration tool. System administrators can modify specific system property information after the Tivoli Identity Manager Server is installed using the system configuration tool.

System properties can be modified at any time. The Tivoli Identity Manager Server may need to be restarted when changes are made to certain system properties. For example, changes to the system properties made by the server startup modules are not recognized unless you restart the server. Changes to other system properties can be recognized within 30 seconds. The most noticeable example is the logging properties. Logging properties can be changed without restarting the server and changes take effect within 30 seconds.

The runConfig utility is a graphical tool used to edit the most commonly used properties for the Tivoli Identity Manager Server. Changes made with this tool are automatically written to the appropriate system property file. The runConfig program is located in the bin directory.

You can view or edit the following system properties with runConfig:

- Application server information
  - Tivoli Identity Manager Server host name (read-only value for WebSphere; editable value for WebLogic)
  - TCP/IP port number (read-only value for WebSphere; editable value for WebLogic)
  - SSL TCP/IP port number (read-only value for WebSphere; editable value for WebLogic)
  - Scheduling information
- Directory repository information
  - Directory Server host name
  - Principal DN and password that the Tivoli Identity Manager Server uses to log onto the directory server
  - Port number for the directory server
  - LDAP connection pool information
- Database repository information
  - Database type
  - Database server IP address and port
  - Database service name
  - Database account and password that the Tivoli Identity Manager Server uses to log onto the database
  - Database connection pool information
- Logging information
  - Level of tracing and verbosity
- Mail notification information
  - The login URL for the Tivoli Identity Manager Server
  - The SMTP mail host to send mail notification
  - The Addressee name for mail notifications
- User interface information
  - Customer logo definition
Property Files

System and supplemental property files are located on the Tivoli Identity Manager Server in the ITIM_HOME/data directory. These files contain all of the system and supplemental properties used by the server.

- For detailed information about system properties (enrole.properties), see Chapter 10, “Configuring System Properties,” on page 91.
- For detailed information about supplemental properties, see Chapter 11, “Configuring Supplemental Properties,” on page 127.

Starting the System Configuration Tool (runConfig)

The system configuration tool is available in both the Windows® and UNIX® versions of Tivoli Identity Manager.

Note: The runConfig utility on UNIX systems (including Linux) requires using the X Window System.

Starting the System Configuration Tool (UNIX)

To start the system configuration tool in UNIX:

1. Log in as root to the system where the Tivoli Identity Manager Server is installed.
2. Change to the Tivoli Identity Manager home directory.
   
   # cd ITIM_HOME

3. Change to the /bin directory.
   
   # cd bin

4. Verify that the current directory is the ITIM_HOME/bin directory.
   
   # pwd
   $ ITIM_HOME/bin

5. Type runConfig and press Enter

   The System Configuration user interface appears.
   You are now ready to change the system properties.

6. Change the desired system properties and click OK.

   See the following sections for more information about configurable system properties.

   Note: Clicking OK saves the changes and closes the user interface. Clicking Apply saves the changes without closing the user interface.

Starting the System Configuration Tool (Windows)

To start the system configuration tool in Windows:

1. Log in as the system administrator account to the operating system where the Tivoli Identity Manager Server is installed.
2. Open Windows Explorer.
3. Open the Tivoli Identity Manager Server home directory.
4. Open the bin directory.
5. Double-click the runConfig program.
   The System Configuration user interface appears.
   You are now ready to change the system properties.
6. Change the desired system properties and click OK.
   See the following sections for more information about configurable system properties.

   **Note:** Clicking OK saves the changes and closes the user interface. Clicking **Apply** saves the changes without closing the user interface.

IBM recommends restarting the Tivoli Identity Manager Server after modifying any property using the system configuration tool.

### General System Properties

The **General** tab of the System Configuration user interface displays configuration fields for Application Server and Scheduling Information.

![General tab](image)

**Figure 1. General tab**

See also, Chapter 10, “Configuring System Properties,” on page 91

### Application Server

**Note:** For WebSphere, the information displayed in this section is for reference only. This information cannot be modified. For WebLogic, these fields can be modified.

The Application Server Information field displays information about the application server machine including the host name, TCP port, and SSL port.
Scheduling Information

Heart Beat
The Scheduling Information field displays information about how frequently a scheduling thread queries scheduled message stores for events to process (Heart Beat). The Heart Beat is measured in seconds. System administrators can only modify the Heart Beat.

Recycle Bin Age Limit
When you delete Tivoli Identity Manager objects (such as organization units, persons, or accounts), the objects are not immediately removed from the system. Instead, they are moved to a recycle bin container. Emptying the recycle bin is a separate process (called “garbage collection”) that involves manually running cleanup scripts.

The Recycle Bin Age Limit field specifies the number of days an object remains in the system’s recycle bin before it becomes available for deletion by manually-run cleanup scripts. The Recycle Bin Age Limit protects objects in the recycle bin from cleanup scripts for the specified length of time.

With the Recycle Bin Age Limit setting, cleanup scripts can only remove those objects that are older than the age limit setting. For example, if the age limit setting is 62 days (default), only objects older than 62 days (have been in the recycle bin for more than 62 days) can be deleted by manually-run cleanup scripts.

You can remove recycle bin entries with expired age limits using the following cleanup scripts:

Windows:

ITIM_HOME\bin\win\ldapClean.cmd

UNIX:

ITIM_HOME/bin/unix/ldapClean.sh

It is recommended that you schedule the recycle bin cleanup process to run periodically. On the Windows platform, you can register the above command script with the Windows scheduler. On the UNIX platform, you can create a UNIX cron job.

A sample UNIX cron script is provided:

ITIM_HOME/bin/unix/schedule_garbage.cron

Directory Properties

The Directory tab of the System Configuration user interface displays Tivoli Identity Manager Server directory connection information and LDAP Connection Pool Information. The Directory tab also has a Test button to test the connection to the directory server.
Tivoli Identity Manager Server Directory Connection Information

Tivoli Identity Manager Server directory connection information fields display the directory server's Principal DN, Password, Host Name, and Port number.

LDAP Connection Pool Information

LDAP Connection Pool Information defines a pool of LDAP connections accessible by Tivoli Identity Manager Server. The Maximum Pool Size field displays the maximum number of connections the LDAP Connection Pool can have at any time. The Initial Pool Size field displays the initial number of connections created for the LDAP Connection Pool. The Increment Count field displays the number of connections added to the LDAP Connection Pool every time a connection is requested once all connections are in use.

**Note:** Once a connection is established and data is stored in the LDAP Directory server, changing the host name or the port number could have detrimental effects.

Database Properties

The **Database** tab of the System Configuration user interface displays general database information and Database Pool information. The **Database** tab also has a **Test** button to test the connection to the database. Depending on the type of connection that is used, one of various dialog windows is displayed when configuring database properties.

**Note:** The database contains the audit trail and workflow information for the system. Changing the configuration after the system is set up can have detrimental effects.

The following dialog window displays the **Database** tab when Tivoli Identity Manager Server does not use an Oracle Client to connect to the Oracle database.
This type of connection uses a Type IV (Oracle Thin) JDBC driver. Similarly, the Microsoft SQLServer uses a type 4 JDBC driver.

![System Configuration](image)

**Figure 3. Database tab**

See also "enRoleDatabase.properties" in Chapter 11, “Configuring Supplemental Properties,” on page 127.

### General Database Information

General database information fields display information such as Database Type, Database Name or Alias, and Database User.

- **The Database Type field** displays the type of database used for your system.
- **DB2 only:** The Database Name or Alias field displays how the Tivoli Identity Manager Server connects to the database.
  - If the database is installed locally, the value represents the name of the database.
  - If the database is installed remotely, the value represents the local alias name of the remote database.
- **Oracle only:** The field label is: Database IP:Port:Name
- **Microsoft SQLServer only:** The field label is: Database IP:Port:Service Name
- **Oracle and Microsoft SQLServer only:** Type 4 JDBC drivers are used. No client software is required.
- **The Database User field** displays the account that the Tivoli Identity Manager Server uses to log in to the database. The user ID must be "enrole", which is created by the Tivoli Identity Manager database configuration program (DBConfig).
  - This account must have a valid user password.
- **The User Password field** is the password for the Database User account.

### Database Pool Information

Database Pool Information determines the number of JDBC connections. The Initial Capacity field displays the initial number of JDBC connections. The Maximum Capacity field displays the maximum number of JDBC connections that Tivoli Identity Manager Server can open to the database at any one time. The Login Delay Seconds field displays the time, in seconds, between connections.
Logging Properties

The Logging tab of the system configuration tool displays the logging and tracing preferences in the Tivoli Identity Manager Server.

![System Configuration Tool](image)

*Figure 4. Database tab*

See also "enRoleLogging.properties" in Chapter 11, "Configuring Supplemental Properties," on page 127

Logging Level

The Tivoli Identity Manager Server logs events in a log file. The Logging Level field displays how verbose the logs are when tracing system errors. System administrators can select how detailed the log file should be by setting the Logging Level field number between INFO and FATAL. FATAL writes less information to the log file than INFO. INFO produces higher amounts of written notifications. For better Tivoli Identity Manager performance, use FATAL.

Trace Exceptions

The Tivoli Identity Manager Server traces system errors. Tracing collects diagnostic information for IBM customer support. System administrators can turn tracing on or off by selecting the Yes or No Trace Exceptions radio buttons.

Mail Properties

The Mail tab of the system configuration tool displays mail notification and gateway parameters.
Web Server Information

The login URL to Tivoli Identity Manager is first presented as a hyperlink in e-mail to new Tivoli Identity Manager users. This login URL is based on the URL value displayed in the Identity Manager Server URL field (base URL) of the Mail tab.

Note that you only specify the host name (or IP address) and port in the base URL. Make sure that it matches the published login URL to your Tivoli Identity Manager system.

For a single-server using the WebSphere Application Server, the base URL should be that of the Web server (for example, the IBM HTTP Server) which by default uses port 80 for HTTP and port 443 for HTTPS (not the ports used by the application server, which by default uses port 9080 for HTTP and port 9443 for HTTPS).

For a cluster and functional cluster using the WebSphere Application Server, the base URL should be that of the Web server which load-balances to all application server instances in the cluster (not the base URL of a specific application server instance).

For a single-server that uses the WebLogic Application Server and does NOT use a commercial Web server, the base URL should be that of the application server, which has a built-in Web server component. For example, 7001 for HTTP and 7002 for HTTPS.

For a single-server that uses the WebLogic Application Server and a commercial Web server, the base URL should be that of the Web server, which by default uses port 80 for HTTP and 443 for HTTPS (not the ports used by the application server).

For a cluster using the WebLogic Application Server, the base URL should be that of the proxy server – either a commercial Web server with the BEA WebLogic plug-in installed and configured, or another WebLogic Application Server configured and running as the proxy server. The base URL should NOT be that of a specific application server instance in the cluster.
Mail Information

The Mail From address refers to the Tivoli Identity Manager system administrator e-mail address for your site.

All e-mail will be delivered from the Mail From parameter. This field is a **required** value. There must be a properly formatted e-mail address in this field.

Mail Server Information

SMTP mail servers are supported. The SMTP host is the mail gateway.

User Interface Properties

The **UI** tab of the system configuration tool allows system administrators to customize the Tivoli Identity Manager Server GUI.

![System Configuration](image)

*Figure 6. UI tab*

See also “UI.properties” in Chapter 11, “Configuring Supplemental Properties,” on page 127

Customer Logo and Customer Logo Link

The Customer Logo field displays the file name of the logo graphic. The Customer Logo Link is an optional URL link activated by clicking on the logo image. System administrators can specify these two variables to replace the IBM logo with their company’s logo throughout the Tivoli Identity Manager system.

Refer to “Configuring a Custom Logo” on page 15 for detailed information about changing the logo and link.

List Page Size

The List Page Size field displays how many items will be displayed on lists throughout the user interface. If the total number of items exceeds the set List Page Size, the list will be spread over multiple pages.

Refer to “Customizing List Displays” on page 15 for detailed information about changing the list page size.
Security Properties

The Security tab of the system configuration tool displays the encryption settings and application server user management preferences in the Tivoli Identity Manager Server.

![System Configuration](image)

*Figure 7. Security tab*

See also "enRoleAuthentication.properties" in Chapter 11, “Configuring Supplemental Properties,” on page 127.

Encryption Settings

**Encryption** (check box)

When checked, the passwords used for database and LDAP connections and the password of the EJB user that is used for EJB authentication are encrypted. The encryption flags are set to true. The flags are represented by the following properties in enRole.properties:

- `enrole.password.database.encrypted`
- `enrole.password.ldap.encrypted`
- `enrole.password.appServer.encrypted`

When the check box is uncheck, the passwords are decrypted and the flags set to false.

Application Server User Management Settings

Enables you to set and confirm the password for the following:

- **System User**
  The WebSphere Application Server user ID and password.
- **EJB User**
  A user and password that you must have defined prior to starting installation.

**Note:** If this field is pre-filled when it appears, the field may contain the value of the System User. Change the field to the value of the EJB user. For more information, refer to the Security Considerations appendix in the appropriate version of the *IBM Tivoli Identity Manager Server Installation Guide*. 
Configuring Properties from the Tivoli Identity Manager GUI

You can also modify certain system properties from within the Configuration section of the Main Menu Navigation Bar in the Tivoli Identity Manager GUI.

From the Configuration tab, you can modify the following properties:

- Lost password question behavior
- Enable/disable password editing
- Password expiration period (number of days)
  - This property is only for Tivoli Identity Manager Server account. The user has to change the password before this period is reached. Whenever new password is set for the Tivoli Identity Manager Server account, the password expiration period will be affected from that time. You can make this period never expire by setting this to zero.
- Password retrieval expiration period (number of hours)
  - After the new account is created, the user will receive an e-mail with the URL link where he/she can get the password. The user has to get the password before this password retrieval period expires.
- Maximum number of invalid logon attempts
  - Sets the maximum number of invalid logon attempts. If exceeded, the account is suspended. The default setting is "0" (unlimited logon attempts).
Chapter 2. Customizing the Tivoli Identity Manager GUI

This chapter contains information on customizing the Tivoli Identity Manager GUI.

See also "UI.properties" in Chapter 11, “Configuring Supplemental Properties,” on page 127.

Section topics:
- “Configuring a Custom Logo” on page 15
- “Customizing List Displays” on page 15
- “Custom Display Attributes” on page 16

Configuring a Custom Logo

The Tivoli Identity Manager GUI can display a company’s logo in the upper right corner of every page. The logo can also be linked to a URL. By default, an IBM logo (IBM_banner.gif) is displayed and the IBM Web site is referenced when the user clicks on the logo image. System administrators can add their company logo by completing the following procedures.

Adding a Logo to the Tivoli Identity Manager GUI

1. Copy a GIF version of the company logo to the following location:
   - **WebSphere:**
     WAS_HOME/installedApps/enrole.ear/enrole.war/images
   - **WebLogic:**
     BEA_HOME/user_projects/itim/applications/enrole/images

2. Start the System Configuration Tool.
   See Chapter 1, “Using the System Configuration Tool (runConfig),” on page 3 for more information.

3. Click the **UI** tab.

4. Type the name of the GIF file in the Customer Logo text field.

5. Optional: Type a URL in the Customer Logo Link text field to link the logo to a Web site.

6. Click **OK**.
   The modifications are updated and saved in the **UI.properties** file and the System Configuration Tool closes.

Customizing List Displays

Many of the pages in the Tivoli Identity Manager GUI list items at a specific level. These list pages can be configured to display a set number of items per page and a set number of links to continuation pages for the lists. By default, a maximum of ten (10) items are listed per page and a maximum of ten (10) links are displayed per page.

These two parameters are configured in two different locations. The number of items per page in a list is set using the System Configuration Tool. The number of links to continuation pages is set in the **UI.properties** file located in the ITIM_HOME/data folder.
Setting the Maximum Number of Items per Page

1. Start the System Configuration Tool.
   See Chapter 1, “Using the System Configuration Tool (runConfig),” on page 3 for more information.
2. Click the UI tab.
3. Type the number of items to display per page in the List Page Size text field.
4. Click OK.
   The modifications are saved and the System Configuration user interface closes.

Setting the Maximum Number of Links per Page

1. Log into the system where the Tivoli Identity Manager Server is installed.
2. Change to the data folder.
3. Open the UI.properties file in a text editor.
4. Change the enrole.ui.pageLinkMax value to the desired value.
   The following is an example of this property in the file:
   ```
enrole.ui.pageLinkMax=10
   ```
5. Save and close the UI.properties file.
   The changed settings appear immediately in the system’s user interface.

Custom Display Attributes

The Tivoli Identity Manager GUI can be customized to use corporate fonts and colors and display limited items per page.

Always restart the system after any changes are made.

Customizing Fonts and Colors

System fonts and colors used by the GUI can be customized by modifying values in the Styles.css file located in the following folder:

**WebSphere:**
```
WAS_HOME/installedApps/<server-name>/enrole.ear/app_web.war/en
```

**WebLogic:**
```
BEA_HOME/user_projects/itim/applications/enrole/en
```
Chapter 3. Configuring Account and Password Management

Section topics:

- “Configuring Suspend/Restore Person with Accounts” on page 17
- “Configuring Optional Password on Person/Account Restore” on page 17
- “Specifying Invalid Passwords” on page 18
- “Excluding Accounts from Reconciliation” on page 18
- “E-mail Template Customization” on page 19

Configuring Suspend/Restore Person with Accounts

The following property can be manually added to the enRole.properties file to modify the behavior of Suspend and Restore Persons when controlled by application APIs:

```
com.ibm.itim.personManagement.suspendRestorePersonWithAccounts
```

Values are:

- **true**
  The Suspend and Restore Person APIs also suspend and restore the Person’s accounts.

- **false**
  The Suspend and Restore Person APIs do not suspend and restore the Person’s accounts.

The default behavior without this property is as follows:

- Suspending a Person also suspends the Person’s accounts
- Restoring the Person does not restore the person’s accounts.

Configuring Optional Password on Person/Account Restore

The following property can be manually added to the enRole.properties file to modify the behavior of password prompting upon Account Restore (and Person Restore if "Restore Accounts" is checked):

```
account.restore.password.suppress
```

This property affects both Account Restore and Person Restore (if "Restore Accounts" is checked).

Values are:

- **true**
  The password prompt for a restored account or Person (with "Restore Accounts" checked) is disabled. Instead, a confirmation window displays.

  However, this behavior does not occur as described if one or more of the selected accounts requires a password for the restore. In this case, Tivoli Identity Manager still prompts for a password even though this property is set to "true" (suppress).

  Note that the agent service profile, in the resource.def file, defines whether a password is required for account restore.
Specifying Invalid Passwords

The Tivoli Identity Manager Server can be configured to prevent users from using specific words as passwords for their accounts. These words are stored in a password dictionary in the LDAP Directory Server. This password dictionary contains a list of words that cannot be used as passwords.

This dictionary can be modified through an LDAP browser by creating `erDictionaryItem` entries under the `erDictionaryName=` entry or by importing an LDIF file with the entries listed into the Directory Server.

The following is an example of an LDIF file with various words to exclude as passwords listed:

```
dn: erword=apple, erdictionaryname=password, ou=ITIM, dc=com
objectClass: top
objectClass: erdictionaryitem
erWord: apple

dn: erword=orange, erdictionaryname=password, ou=ITIM, dc=com
objectClass: top
objectClass: erdictionaryitem
erWord: orange
```

The only value that must be modified is the `erWord` value. The `erWord` value specifies the word that is not allowed to be used as a password.

**Note:** The final line in the LDIF file must be followed by a carriage return for the entry to be recognized.

Once the password dictionary is populated with the desired words, the password policies must be modified to use the dictionary. Refer to the *IBM Tivoli Identity Manager Policy and Organization Administration Guide* for more information on modifying password policies.

Adding Words to the Password Dictionary

To add words to the password dictionary, do the following:

1. Create an LDIF file specifying the words to add to the password dictionary.
2. Import the LDIF file to the LDAP Directory Server.

Excluding Accounts from Reconciliation

During reconciliations, all accounts are returned from the managed resource, unless otherwise specified by a query. Accounts are automatically adopted if the account is owned by a recognized user in the system or if an alias exists for any account.

However, the Tivoli Identity Manager Server can be configured to prevent automatic adoption of specified accounts. This feature can be used to prevent system accounts, such as `root`, `lp`, `sys`, and `etc` in UNIX resources, from automatically being adopted. This prevents users from accidentally or maliciously adopting and modifying sensitive accounts.

Although these accounts are not automatically adopted, these accounts can still be manually adopted by an administrative user.
The accounts to exclude from reconciliations are specified in an LDIF file. The following is an example of entries in an LDIF file:

dn: ou=excludeAccounts, ou=ITIM, ou=ITIM, dc=com
ou: excludeAccounts
objectClass: top
objectClass: organizationalunit
dn: cn=SolarisProfile, ou=excludeAccounts, ou=ITIM, ou=ITIM, dc=com
erObjectProfileName: SolarisProfile
objectClass: top
objectClass: eridentityexclusion
cn: SolarisProfile
erAccountID: root
erAccountID: admin

The cn and erObjectProfileName is the name of the service profile. Excluded accounts are defined by the erAccountID attribute. The example excludes the root and admin accounts from automatically being adopted when a reconciliation is run on a Solaris service.

Selecting Accounts to Exclude from Reconciliations

To select accounts to exclude from reconciliations, do the following:

1. Create an LDIF file specifying the accounts to exclude from reconciliation and the services on which these accounts exist.
2. Import the LDIF file to the LDAP Directory Server.

See also "Reconciliation Information" in Chapter 11, "Configuring Supplemental Properties," on page 127

E-mail Template Customization

The Tivoli Identity Manager Server uses a static hypertext markup language (HTML) file as a generic template for all standard e-mail messages regarding workflow processes within the system. This file controls the information is displayed in the e-mail message and how the information is displayed.

The HTML file is named notifytemplate.html and is located in the ITIM_HOME/data/workflow_systemprocess directory.

Other than the three variables prefaced by a dollar sign ('$'), all other attributes in the HTML file can be modified. The three variables are:

• $TITLE
• $BODY
• $BASE_URL

Common attributes to modify are the background color and table attributes such as cell size and color.
# Part 2. Advanced System Configuration

## Chapter 4. Configuring SSL Communication

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of SSL and Digital Certificates</td>
<td>23</td>
</tr>
<tr>
<td>Understanding Private Keys and Digital Certificates</td>
<td>23</td>
</tr>
<tr>
<td>Key Formats</td>
<td>24</td>
</tr>
<tr>
<td>SSL Implementation Types</td>
<td>24</td>
</tr>
<tr>
<td>Configuration Summary and Roadmap</td>
<td>26</td>
</tr>
<tr>
<td>Tivoli Identity Manager Deployment on WebSphere</td>
<td>26</td>
</tr>
<tr>
<td>Tivoli Identity Manager Deployment on WebLogic</td>
<td>26</td>
</tr>
<tr>
<td>Configuring Browser-to-Web-Server SSL (WebSphere)</td>
<td>27</td>
</tr>
<tr>
<td>1. Generating a Certificate Signing Request (CSR)</td>
<td>27</td>
</tr>
<tr>
<td>2. Installing the Signed Certificate</td>
<td>28</td>
</tr>
<tr>
<td>3. Configuring the Web Server for SSL</td>
<td>29</td>
</tr>
<tr>
<td>Configuring Browser-to-Web-Server SSL (WebLogic)</td>
<td>31</td>
</tr>
<tr>
<td>Configuring Server-to-Agent SSL</td>
<td>33</td>
</tr>
<tr>
<td>Configuring the Server Certificates for One-way SSL</td>
<td>34</td>
</tr>
<tr>
<td>Using the Root CA Certificate of a Third-party CA</td>
<td>34</td>
</tr>
<tr>
<td>Using Certificates Generated and Signed with OpenSSL Utilities</td>
<td>34</td>
</tr>
<tr>
<td>Configuring the Signed Certificate on the Agent</td>
<td>35</td>
</tr>
<tr>
<td>Configuring Agent-initiated SSL (Agent-to-Web-Server)</td>
<td>36</td>
</tr>
<tr>
<td>ADK-based Agent Configured for Event</td>
<td>36</td>
</tr>
<tr>
<td>Identity Feed for a Program using the JNDI Interface</td>
<td>36</td>
</tr>
<tr>
<td>IBM Directory Integrator (IDI)-based Agent</td>
<td>36</td>
</tr>
</tbody>
</table>

## Chapter 5. Configuring Single Sign-on Solutions

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of Single Sign-on Capability</td>
<td>39</td>
</tr>
<tr>
<td>Configuring Single Sign-on with WebSEAL</td>
<td>40</td>
</tr>
<tr>
<td>Prerequisites, Warnings, and Workarounds</td>
<td>40</td>
</tr>
<tr>
<td>Warning: SSO with Multiple Person Accounts</td>
<td>40</td>
</tr>
<tr>
<td>Prerequisite: Setting the Language Environment:</td>
<td>40</td>
</tr>
<tr>
<td>Workaround: Configuring the Time Zone Offset</td>
<td>40</td>
</tr>
<tr>
<td>Configuration Procedure</td>
<td>41</td>
</tr>
<tr>
<td>Configuring Single Sign-on with Tivoli Access</td>
<td>42</td>
</tr>
<tr>
<td>Manager Plug-in Servers</td>
<td>42</td>
</tr>
<tr>
<td>Creating a WebSEAL Junction with Tivoli Identity Manager</td>
<td>42</td>
</tr>
<tr>
<td>Creating a TCP Junction</td>
<td>42</td>
</tr>
<tr>
<td>Creating an SSL Junction</td>
<td>45</td>
</tr>
<tr>
<td>Specifying the URL for a Junction</td>
<td>48</td>
</tr>
</tbody>
</table>
Chapter 4. Configuring SSL Communication

This chapter describes the administration and configuration tasks required to set up a Tivoli Identity Manager deployment to use digital certificates for authentication over SSL.

Overview and Configuration Summary:

- “Overview of SSL and Digital Certificates” on page 23
- “Configuration Summary and Roadmap” on page 26

Configuration Details for Tivoli Identity Manager on WebSphere:

- “Configuring Browser-to-Web-Server SSL (WebSphere)” on page 27
- “Configuring Server-to-Agent SSL” on page 33

Configuration Details for Tivoli Identity Manager on WebLogic:

- “Configuring Browser-to-Web-Server SSL (WebLogic)” on page 31
- “Configuring Server-to-Agent SSL” on page 33

Supplementary information:

- “Configuring Agent-initiated SSL (Agent-to-Web-Server)” on page 36

Overview of SSL and Digital Certificates

A Tivoli Identity Manager deployment must consider the security of communication between all configured components. The industry-standard Secure Sockets Layer (SSL) mechanism, which uses digital certificates for authentication, is used for secure communication in a Tivoli Identity Manager deployment.

SSL provides secure connections by allowing two applications connecting over a network connection to authenticate each other’s identity. Additionally, SSL provides encryption of the data exchanged between the applications. Authentication allows a server (one-way) and optionally a client (two-way) to verify the identity of the application on the other end of a network connection. Encryption makes data transmitted over the network intelligible only to the intended recipient.

Features of SSL include the following concepts:

- SSL provides a mechanism for one application to authenticate itself to another application.
- One-way SSL allows one application to be certain of the identity of the other application.
- Two-way SSL (mutual authentication) allows both applications to be certain of the identity of each other.
- The application that assumes the “server” role possesses and uses a server-side certificate to prove its identity to the client application.
- In mutual authentication, the application that assumes the “client” role possesses and uses a client-side certificate to prove its identity to the server application.
• The application that is presented with a certificate must have in its possession the root certificate (or certificate chain) of the Certificate Authority (CA) that signed the certificate being presented. The root CA certificate, or chain, validates the certificate being presented.
• In client connections, the client browser alerts the user when presented with a certificate that is not issued by a recognized Certificate Authority.

Understanding Private Keys and Digital Certificates

Private keys, digital certificates, and trusted Certificate Authorities can be used to establish and verify the identity of network applications.

SSL uses public key encryption technology for authentication. In public key encryption, a public key and a private key are generated for an application. The keys are related such that data encrypted with the public key can only be decrypted using the corresponding private key. Similarly, the data encrypted with the private key can only be decrypted using the corresponding public key. The private key is carefully protected so that only the owner can decrypt messages that were encrypted using the public key.

The public key is embedded into a digital certificate with additional information describing the owner of the public key, such as name, street address, and e-mail address. A private key and digital certificate provide identity for the application.

The data embedded in a digital certificate is verified by a trusted Certificate Authority (CA) and digitally signed with the Certificate Authority’s digital certificate. Well-know Certificate Authorities include Verisign and Entrust.net. A trusted Certificate Authority establishes trust for an application.

An application participating in an SSL connection is authenticated when the other party evaluates and accepts their digital certificate. A digital certificate used to authenticate is validated by an associated root CA certificate located on the receiving application.

Web browsers, servers, and other SSL-enabled applications generally accept as genuine any digital certificate that is signed by a trusted Certificate Authority and is otherwise valid. For example, a digital certificate can be invalidated because it has expired or the digital certificate of the Certificate Authority used to sign it expired. A server certificate can be invalidated if the host name in the digital certificate of the server does not match the host name specified by the client.

Key Formats

WebLogic Server can use digital certificates in either .pem, .arm, or .der format.

A .pem (privacy-enhanced mail) format file begins and ends with the following lines:
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----

A .pem file format supports multiple digital certificates (for example, a certificate chain can be included). However, the ordering of the digital certificates in the file is important. For example, cert A, cert B (must be the issuer of cert A), cert C (must be the issuer of cert B),... to the root CA.
A .arm file format is an ASCII encoded binary file. A .arm file contains a base-64 encoded ASCII representation of a certificate, including its public key, but not its private key. A .arm file format is generated and used by the GSKit iKeyman utility (WebSphere only).

A .der file format contains binary data. A .der file can be used only for a single certificate, however, a .pem file can be used for multiple certificates.

**SSL Implementation Types**

Components in a Tivoli Identity Manager configuration make use of several implementations of SSL:

- IBM Global Security Toolkit (GSKit)
  - Used by the IBM HTTP Server.
- RSA SSL-J
  - Used by the Tivoli Identity Manager.
- RSA SSL-C
  - Used by the Agent.
- Open SSL
  - Used by the Agent.

A Tivoli Identity Manager configuration uses secure SSL communication in three locations:

- Between the client user’s Web browser and the Web Server used by the Tivoli Identity Manager application server
- Between the Tivoli Identity Manager Server and the Tivoli Identity Manager agent(s)
- Agent-initiated SSL between the Web Server and an Agent
Configuration Summary and Roadmap

The information in this section provides a roadmap for configuring your Tivoli Identity Manager deployment for SSL.

**Tivoli Identity Manager Deployment on WebSphere**

The following configuration summary pertains to a Tivoli Identity Manager deployment on the WebSphere application server.

- Configure SSL between the client browser and the Web server (IBM HTTP Server).
  - Only one-way authentication is supported.
  - See “Configuring Browser-to-Web-Server SSL (WebSphere)” on page 27

- Configure SSL between the Tivoli Identity Manager Server and the agent(s).
  - Only one-way authentication is supported.
  - See “Configuring Server-to-Agent SSL” on page 33

**Tivoli Identity Manager Deployment on WebLogic**

The following configuration summary pertains to a Tivoli Identity Manager deployment on the WebLogic application server.

- Configure SSL between the client browser and the built-in WebLogic Web server.
  - Only one-way authentication is supported.
  - See “Configuring Browser-to-Web-Server SSL (WebLogic)” on page 31

- Configure SSL between the Tivoli Identity Manager Server and the agent(s).
  - Only one-way authentication is supported.
  - See “Configuring Server-to-Agent SSL” on page 33
Configuring Browser-to-Web-Server SSL (WebSphere)

The following information pertains to a Tivoli Identity Manager deployment on the WebSphere application server.

**Deployment summary:**
- The client is a user with a Web browser.
- The Web server is the IBM HTTP Server deployed on a remote machine or, alternatively, on the same machine as WebSphere.
- GSKit implementation of SSL is used by the IBM HTTP Server.

**Note:** In the diagram below, “ITIM Server” refers to the IBM Tivoli Identity Manager Server.

![Diagram of Tivoli Identity Manager deployed on WebSphere](image)

**Figure 8. Tivoli Identity Manager deployed on WebSphere**

**Procedure summary:**
1. Generate a Certificate Signing Request (CSR) that is sent to the Certificate Authority (CA), who then returns a signed certificate used by the Web server to identify itself to the client’s browser.
2. Install the certificate on the Web server.
3. Configure the Web server for SSL.
4. Ensure that the browser has the root certificate of the CA that signed the Web server’s certificate. The CA root certificate is used by the browser to validate the certificate sent by the Web server. Industry-standard root CA certificates (such as VeriSign) are typically part of the browser distribution.

**1. Generating a Certificate Signing Request (CSR)**

To acquire a certificate from a Certificate Authority (CA), you must first submit a Certificate Signing Request (CSR).

The WebSphere Application Server has a Java application that can generate certificate requests. This tool is called **iKeyman**. **iKeyman** is a servlet that collects information from your system and generates a private key file and a certificate request file. The servlet lets you submit the CSR file to a CA (such as VeriSign) for signing.
For detailed information about using the WebSphere Application Server utilities, refer to the WebSphere Application Server documentation library at the following Web site:


The following procedures describe how to generate a CSR:
1. Start the WebSphere Application Server key management utility iKeyman. Find and execute gsk5ikm in the ../.gsk5 directory.
2. Create a keyfile or open an existing key database file.
3. Select the Signer Certificates or Personal Certificates menu and click Personal Certificate Requests.
4. Click New.
5. Complete the following fields:
   • Key Label
   • Common Name
   • Organization
   • File Name
6. Click OK.
   A dialog window appears stating that the certificate request is generated and stored in the previous specified file.
7. Click OK.
   The dialog window closes.
8. Exit iKeyman.
9. Submit the certificate request to the appropriate CA.

2. Installing the Signed Certificate
The discussion in this section assumes you have received a signed certificate from the Certificate Authority. Additionally, you have saved this certificate file in a temporary directory.

The following procedures describe how to install a certificate using WebSphere Application Server utilities.

For detailed information about using the WebSphere Application Server utilities, refer to the WebSphere Application Server documentation library:


1. Start the WebSphere Application Server key management utility iKeyman.
2. Open the database file used to create the certificate request.
3. Click the Personal Certificate Requests menu and click Personal Certificates.
4. Click Receive.
5. Click Data Type and select the data type of the digital certificate.
   • For ASCII-formatted certificates, select the “Base64-encoded ASCII Data” data type.
   • For binary-formatted certificates, select the "Binary DER Data" data type.
6. Specify the temporary directory location and file name of the certificate.
7. Click OK.
8. Type the label of the new digital certificate and click OK.
iKeyman saves the certificate in the key database file and the certificate is listed in the Personal Certificates list.


Note: If the Web server uses a self-signed certificate instead of a certificate issued by a CA such as VeriSign, then the client browser prompts the user to decide whether to trust the unknown signer of the server’s certificate.

3. Configuring the Web Server for SSL

Once you have the certificate installed, you must configure the IBM HTTP Server for SSL.

1. Start the WebSphere Application Server key management utility iKeyman.
2. Create a keyfile to store the SSL key files and certificate. For example:
   
   `ITIM_HOME/myKeys`

3. Click the Key Database File menu and select New.
4. Define the settings as listed below and click OK.
   - **Key Database Type**: CMS Key Database File
   - **File Name**: WebServerKeys.kdb
   - **Location**: path to `ITIM_HOME/myKeys` directory

5. Enter a password for your SSL key file and confirm the password.
6. Select the Stash the password to a file? check box.
7. Click OK.

   This causes a file named `WebServerKeys.sth` to be created containing an encoded form of the password.

   **Note**: Operating system permissions should be used to prevent all access to this file by unauthorized persons.

8. Open the Signer Certificates menu and select Personal Certificates when you see the list of default Signer Certificates.

   - If you have a server certificate from a CA (for example, VeriSign), you can click Import to import this certificate into your SSL key file. You will be prompted for the type and location of the file containing the server certificate.
   - If you do not have a valid server certificate from a CA, but want to test your system, click New Self-Signed.

   You will be prompted to enter a Key Label, such as ITIM, and an Organization, such as IBM. Use the default values for all other values.

9. Open the Key Database File menu and select Close.

10. Add the following lines to the bottom of your `httpd.conf` file (substituting `ITIM_HOME` with the correct path to your `myKeys` directory):

    ```
    LoadModule ibm_ssl_module libexec/mod_ibm_ssl_128.so
    Listen 443
    SSLEnable
    Keyfile "ITIM_HOME/myKeys/WebServerKeys.kdb"
    ```

    This causes the Web server to listen on port 443 (the default SSL port).

11. Add ports 443 and 9443 to the VirtualHost on the WebSphere Server administration console on the PRIMARY system and update the Web server plugin.

12. Start the IBM HTTP Server:

    - **Solaris**: `/opt/IBMHttpServer/bin/apachectl start`
    - **Linux**: `/opt/IBMHttpServer/bin/apachectl start`
13. Test the configuration from a browser by entering a URL. For example:
https://localhost

**Note:** If you are using a self-signed certificate, rather than a certificate issued by a Certificate Authority such as VeriSign, then your browser prompts you to see if you want to trust the unknown signer of the Web server’s certificate.
Configuring Browser-to-Web-Server SSL (WebLogic)

The following information pertains to a Tivoli Identity Manager deployment on the WebLogic application server.

Deployment summary:
- The client is a user with a Web browser.
- The Web server is built-in with WebLogic.
- RSA SSL-J implementation of SSL is used by WebLogic.

Note: In the diagram below, "ITIM Server" refers to the IBM Tivoli Identity Manager Server.

![Diagram of Tivoli Identity Manager deployed on WebLogic]

Figure 9. Tivoli Identity Manager deployed on WebLogic

Procedure summary:
1. Generate a Certificate Signing Request (CSR) that is sent to the Certificate Authority (CA), who then returns a signed certificate used by the Web server to identify itself to the client’s browser.
2. Install the certificate on the Web server.
3. Configure the Web server for SSL.
4. Ensure that the browser has the root certificate of the CA that signed the Web server’s certificate. The CA root certificate is used by the browser to validate the certificate sent by the Web server. Industry-standard root CA certificates (such as VeriSign) are typically part of the browser distribution.

Conditions:
- Certificates (including the root CA certificate) and keyfiles can be in Base64 encoded ASCII format (.pem) or binary format (.der).
- Use the Certificate Request Generator servlet to obtain private keys, digital certificates, and trusted CA certificates. The servlet is part of the WebLogic distribution.
- Private keys and trusted CA certificates are stored as files in the domain directory.
Detailed procedures:

For complete detailed information on setting up browser-to-Web-server SSL on WebLogic, refer to the BEA WebLogic Web site:

http://e-docs.bea.com/wls/docs70/secmanage/ssl.html
Configuring Server-to-Agent SSL

The following information pertains to a Tivoli Identity Manager deployment on either the WebSphere or the WebLogic application server.

In this scenario, the Tivoli Identity Manager Server initiates communication over SSL with the agent (server-to-agent) to complete a transaction originally initiated by the browser.

There are additional scenarios where the agent initiates communication over SSL with the Tivoli Identity Manager Server (agent-to-server). For further information about these scenarios, refer to “Configuring Agent-initiated SSL (Agent-to-Web-Server)” on page 36.

Deployment summary:
- The Tivoli Identity Manager Server and the agent use one-way authentication over SSL by default.
- RSA SSL-C or Open SSL implementation of SSL is used by the Agent
- RSA SSL-J implementation of SSL is used by the Tivoli Identity Manager Server

Note: In the diagrams below, "ITIM Server” refers to the IBM Tivoli Identity Manager Server.

![Diagram of ITIM Application Server, WebSphere or WebLogic, ITIM Server, CA, Cert A, One-way SSL, and Agent Resource]

Figure 10. Configuring one-way server-to-agent SSL

Procedure summary (default one-way authentication):
1. Generate a Certificate Signing Request (CSR) that is sent to the Certificate Authority (CA), who then returns a signed certificate used by the agent to identify itself to the Tivoli Identity Manager Server.
2. Install the signed certificate on the agent.
3. Ensure the root certificate of the CA that signed the agent’s certificate resides on the Tivoli Identity Manager Server. The CA root certificate is used by the server to validate the certificate sent by the agent.
The Tivoli Identity Manager agents use a utility called CertTool to request, install, delete, and register certificates.

**Configuring the Server Certificates for One-way SSL**

**Conditions for one-way SSL:**
- The Tivoli Identity Manager Server is pre-configured to support one-way authentication over SSL.
- The agent authenticates with a signed certificate; the Tivoli Identity Manager Server verifies that certificate with the associated CA root certificate.
- Certificates (including the root CA certificate) and keyfiles must be in binary format (.der).

**Summary of configuration choices:**
- "Using the Root CA Certificate of a Third-party CA” on page 34
- "Using Certificates Generated and Signed with OpenSSL Utilities” on page 34

**Using the Root CA Certificate of a Third-party CA**

1. Use the agent’s CertTool utility to generate a Certificate Signing Request (CSR) that is sent to the CA.
   
   See "Configuring the Signed Certificate on the Agent” on page 35 for information on configuring certificates for the agent.

2. Use the agent’s CertTool utility to install the signed certificate on the agent.
   
   See "Configuring the Signed Certificate on the Agent” on page 35 for information on configuring certificates for the agent.

3. Manually copy the associated root CA certificate to the ITIM_HOME/cert directory of the Tivoli Identity Manager Server.

**Using Certificates Generated and Signed with OpenSSL Utilities**

You can get started quickly using SSL for server-to-agent communications using the freely available OpenSSL utilities to generate certificates and sign certificate signing requests (CSR’s). These utilities are available at www.openssl.org or can be found installed by default in most Linux distributions. The following steps were verified with version 0.9.6b of the OpenSSL utilities.

1. Generate a CSR with CertTool. Use menu option A, "Generate private key and certificate request.”

2. Enter the appropriate values for the certificate request.

3. Save to a file called agentreq.pem. This file is located in the AGENT_HOME/bin directory.
   
   The private key is written to the registry and the CSR is contained within the .pem file.

4. Copy the agentreq.pem file to the machine used to generate the certificates.

5. To create the certificate authority (CA) private key and signed certificate, perform the following steps at the command line of the certificate machine:
   
   $ openssl
   
   OpenSSL> req -x509 -newkey rsa:1024 -keyout cakey.pem -out cacert.pem
   
   # enter values for CA cert, including country, state, etc.
   
   OpenSSL> quit

6. To set up the signing environment enter:
   
   $ mkdir demoCA
   
   $ cp cacert.pem demoCA/cacert.pem
   
   $ mkdir demoCA/private
   
   $ mv cakey.pem demoCA/private/cakey.pem
$ mkdir demoCA/newcerts
$ touch demoCA/index.txt
$ cat > demoCA/serial
01

7. To sign certificate request enter:
   $ openssl
   OpenSSL> ca -in ntagentreq.pem -out agentcert.pem
   OpenSSL> quit
   $ mv demoCA/newcerts/01.pem agentcert.pem

8. To translate the CA cert to binary format enter:
   $ openssl
   OpenSSL> x509 -inform PEM -outform DER -in demoCA/cacert.pem -out cacert.der
   OpenSSL> quit
   $

9. Copy the signed certificate request back to the agent machine.

10. In CertTool, enter option B, "Install certificate from file".

11. Enter the location of the signed request.

12. Verify that the certificate was installed by viewing the currently installed certificate (CertTool option D).

13. Copy the binary form of the CA certificate file (cacert.der) to the Tivoli Identity Manager Server machine in the ITIM_HOME/cert directory.

**Configuring the Signed Certificate on the Agent**

Refer to the "Certificate Installation" chapter of the appropriate Tivoli Identity Manager Agent Installation Guide for complete information on using the CertTool utility to manage the signed certificate used by the agent for one-way authentication.
Configuring Agent-initiated SSL (Agent-to-Web-Server)

Normally, SSL communication is initiated with the agent (server-to-agent) to complete a transaction originally initiated by the browser.

There are three scenarios where the agent initiates communication over SSL with the Web Server (agent-to-Web-server):

- “ADK-based Agent Configured for Event Notification” on page 36
- “Identity Feed for a Program using the JNDI Interface” on page 36
- “IBM Directory Integrator (IDI)-based Agent” on page 36

In all cases, you must export the root CA certificate used by the Web server using the IBM HTTP Server iKeyman tool. The CA certificate must then be placed in the appropriate key store used by the external agent.

ADK-based Agent Configured for Event Notification

For event notification, an ADK-based agent initiates communication with the Web Server to notify it of new, modified or deleted accounts.

Use the CertTool to install on the agent the root certificate of the CA that signed the Web Server’s certificate. The CA certificate must be in binary format (the IBM HTTP Server iKeyman tool uses a .der file name suffix).

The ADK-based agent uses the registry for the key store. Use the CertTool utility to manage the keys.

Identity Feed for a Program using the JNDI Interface

For identity feed, an external Java program is used instead of an Agent to initiate communication with the Web Server to add new, modify, or delete personal information records. This Java program uses the Java Naming Directory Interface (JNDI) to communicate with the Web Server.

The identity feed program uses the operating system file management utilities to place the root CA certificate in a folder that is specified in the JNDI program. The CA certificate must be in binary format (the IBM HTTP Server iKeyman tool uses a .der file name suffix).

IBM Directory Integrator (IDI)-based Agent

For either identity feed or an account management agent, IDI (IBM Directory Integrator) initiates communication to send data to the Tivoli Identity Manager Server. Alternatively, a custom Java program using the DSMLv2 JNDI provider may do the same thing.

For an IDI-based agent, the root CA certificate should be placed in a Java key store. Use the iKeyman tool provided with the IBM HTTP Server. The CA certificate may be in binary or ASCII base 64 encoded format. The IBM HTTP Server iKeyman tool uses the .der file name suffix for binary format and the .arm file name suffix for ASCII base 64 encoded format.

The Java key store can be created with the iKeyman tool provided with the WebSphere Application Server, or with the Sun Java keytool key and certificate management utility.
For details on the **keytool** utility, refer to the documentation shipped with the Sun JDK, or online at:

http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html

The IDI-based agent uses the Java Key Store (JKS) for the key store.
Chapter 5. Configuring Single Sign-on Solutions

This chapter discusses the single sign-on capabilities for Tivoli Identity Manager.

Section topics:
- “Overview of Single Sign-on Capability” on page 39
- “Configuring Single Sign-on with WebSEAL” on page 40
- “Configuring Single Sign-on with Tivoli Access Manager Plug-in Servers” on page 42
- “Creating a WebSEAL Junction with Tivoli Identity Manager” on page 42

Overview of Single Sign-on Capability

Web security servers provided by IBM Tivoli Access Manager (Tivoli Access Manager) can enable single sign-on capability to Tivoli Identity Manager. This allows Tivoli Access Manager to perform user authentication and coarse-grained authorization before access is allowed to the Tivoli Identity Manager application. The Tivoli Identity Manager application then applies its own fine-grained access control using its own Access Control Information (ACI).

When configured for single sign-on, the Tivoli Identity Manager Server identifies the user by means of an HTTP header variable named "iv-user". Each of the Web security servers provided by Tivoli Access Manager has the ability to securely set this HTTP header before passing traffic to the Tivoli Identity Manager Server.

Notes:

1. When the Tivoli Identity Manager Server is configured for single sign-on, users should not have direct network access to the Tivoli Identity Manager Server. A security exposure exists if a user could create a request with a false value for "iv-user" and send the request directly to the Tivoli Identity Manager Server. For optimum security, it is strongly recommended that network access to the Tivoli Identity Manager Server be possible only through the use of an Access Manager Web server.

2. Both the Tivoli Identity Manager Server and the Tivoli Access Manager Web security server maintain a session with the browser. The Tivoli Identity Manager Server session must have a timeout value that is less than the timeout value for the Tivoli Access Manager Web security server. This is especially applicable to a shared workstation environment.

3. The Tivoli Identity Manager application is "stateful". That is, some operations require several requests to complete, and the results of a request depend upon the data entered in previous requests. If a load-balancing mechanism is used to distribute users across multiple deployments of the Tivoli Identity Manager application, it is necessary to ensure that all requests from the same user are sent to the same Tivoli Identity Manager Server instance.

4. When the Tivoli Identity Manager Server is configured for single sign-on, all JAAS logins will succeed as long as the specified user ID exists. Thus, applications that use the Tivoli Identity Manager API should also be run from a protected location.

Do the following:

1. Modify the Tivoli Identity Manager properties file to enable single sign-on.
2. Configure the Tivoli Access Manager Web security server to insert the user’s identity in traffic that is passed to Tivoli Identity Manager.

3. Apply Tivoli Access Manager authorization to the Tivoli Access Manager protected object space so that only authorized users are allowed access to Tivoli Identity Manager.

Configuring Single Sign-on with WebSEAL

Using WebSEAL authentication in place of Tivoli Identity Manager authentication enables users to enter a single user ID and password on the WebSEAL logon page to gain access to Tivoli Identity Manager. The Tivoli Identity Manager logon panel does not appear during the single sign-on operation.

Prerequisites, Warnings, and Workarounds

Warning: SSO with Multiple Person Accounts
The single sign-on functionality between Tivoli Access Manager and Tivoli Identity Manager is designed to provide identity mapping between a single Tivoli Access manager user and a single Tivoli Identity Manager user. The Tivoli Access Manager user ID is used to find the Tivoli Identity Manager Person owning an account in the Tivoli Identity Manager environment. However, it is possible for a Tivoli Identity Manager Person to have multiple accounts. In this case, the identity mapping to one of the multiple accounts is unpredictable. Use caution when making single sign-on available between Tivoli Access Manager users and Tivoli Identity Manager Persons.

Prerequisite: Setting the Language Environment:
In a single sign-on to WebSEAL environment, the “Select another language” link located on the Tivoli Identity Manager logon panel is not available because the Tivoli Identity Manager logon panel does not appear after the WebSEAL logon.

The workaround to this problem is to set the language environment through the browser before logging on. The following example is appropriate for Microsoft Internet Explorer:

1. Ensure that Encoding is set to Unicode (UTF-8):
   
   View > Encoding > Unicode (UTF-8)

2. Configure the appropriate language:
   
   Internet Options > General > Languages

3. Ensure that the language supports unicode:
   
   Internet Options > General > Fonts

Workaround: Configuring the Time Zone Offset
When logging on to Tivoli Identity Manager server through a WebSEAL junction (single sign-on), the Effective Date field of the GUI displays the server’s time in GMT (Greenwich Mean Time) instead of the local browser time. During single sign-on, the Tivoli Identity Manager logon page, by design, is by-passed. When the logon page is by-passed, the correct time and time zone is not conveyed to the Tivoli Identity Manager server.

As an administrator, you can provide a workaround to this problem:

1. Calculate the time zone offset based on the GMT time zone of the client browser. For example, for Tokyo (Japan), the time zone offset is 9. For California (United States), the time zone offset is -8 (-7 for Pacific Daylight Time).
2. Embed this offset value in the URL that you provide (publish) to your users in the following format:

   \textit{http://WebSEAL-system-address/junction-name/enrole/logon?timezoneOffset=calculated-offset}

   The URL link passes the offset value as an HTTP parameter when submitting the post.

   For example, if the server is in California (United States) and the users (client browsers) are in Tokyo (Japan), publish the following URL to the users in Japan:

   \textit{http://WebSEAL-system-address/junction-name/enrole/logon?timezoneOffset=9}

   Alternatively, you can publish the URL of a Web page on the Web server used by WebSEAL, or a Web page that is part of your corporate portal. The Web page presents the link to the WebSEAL SSO. The page should contain a JavaScript function that calculates the time zone offset between the client browser and GMT. Once clicked, the link should submit an HTTP request (with the auto-calculated time zone offset value passed as an HTTP parameter) to the Tivoli Identity Manager server. This is similar to the URL workaround described above, except that the offset value is automatically calculated.

\textbf{Configuration Procedure}

1. Configure WebSEAL before configuring Tivoli Identity Manager:
   \begin{itemize}
   \item Pass all domain attributes in cookie headers
   \item Recognize UTF-8 encoded strings only
   \end{itemize}

2. Enable WebSEAL single sign-on functionality by setting the \texttt{enrole.ui.ssoEnabled} property to \texttt{TRUE} in the \texttt{ui.properties} file. Tivoli Identity Manager will not display a logon page.

   \texttt{enrole.ui.ssoEnabled = true}

3. By default, the Java object that provides the mechanism for WebSEAL single sign-on is specified in the \texttt{enRoleAuthentication.properties} file (no action required):

   \texttt{enrole.authentication.provider.webseal =}
   \begin{verbatim}
   factory = com.ibm.enrole.authentication.webseal.WebsealProviderFactory
   \end{verbatim}

4. In the \texttt{enRoleAuthentication.properties} file, you must indicate the appropriate algorithm for mapping the Tivoli Access Manager user ID to a Tivoli Identity Manager user ID:

   \begin{itemize}
   \item If the Tivoli Access Manager user ID is the same as the Tivoli Identity Manager user ID:
     \texttt{enrole.authentication.idsEqual = true}
   \item If the Tivoli Access Manager user ID is NOT the same as the Tivoli Identity Manager user ID:
     \texttt{enrole.authentication.idsEqual = false}
   \end{itemize}

   An internal identity mapping algorithm is used to ensure the success of the single sign-on operation.

5. Changing the Tivoli Identity Manager timeout session value in the deployment descriptor may create a security exposure in a shared workstation environment. Set the default value of the Tivoli Identity Manager timeout session value to the following to avoid the security exposure:

   \begin{itemize}
   \item Tivoli Identity Manager will timeout due to inactivity.
   \item Tivoli Identity Manager will timeout at the same time or before a WebSEAL timeout due to inactivity.
   \end{itemize}
6. Configure either a TCP or an SSL junction with Tivoli Identity Manager. For more information, see "Creating a WebSEAL Junction with Tivoli Identity Manager."

Configuring Single Sign-on with Tivoli Access Manager Plug-in Servers

Using Tivoli Access Manager plug-in authentication enables users to enter a single user ID and password to gain access to Tivoli Identity Manager. The plug-in can be either the Tivoli Access Manager plug-in for Web servers or the plug-in for Edge servers.

Do the following:

1. Configure the Tivoli Access Manager plug-in to insert the authenticated user’s identity in the "iv-user" HTTP header. For more information, refer to descriptions of the single configuration model in the Tivoli Access Manager documentation for Web server or the Edge server.

2. Configure the Tivoli Identity Manager ui.properties file and, optionally the enRoleAuthentication.properties file.
   - Set the enrol-ui.ssoEnabled property to TRUE in the ui.properties file; Tivoli Identity Manager will not display a logon page.
   - Set the enrol.authentication.idsEqual property to FALSE in the enRoleAuthentication.properties file if a user’s Tivoli Access Manager userid is not always equal to the user’s Tivoli Identity Manager userid.

Creating a WebSEAL Junction with Tivoli Identity Manager

This section describes creating a WebSEAL junction that uses either a TCP or SSL connection.

Prior to configuring a junction, you must configure WebSEAL for the following:
- Pass all domain attributes in cookie headers
- Recognize UTF-8 encoded strings only

Creating a TCP Junction

To create a TCP junction with Tivoli Identity Manager, do the following:

Note: The following steps assume WebSEAL is already installed and configured. For more information, refer to the WebSEAL installation documentation.

1. Type pdadmin at a command prompt to start the pdadmin command line interface.
2. Type login at the pdadmin command prompt and log in with the security master’s user name and password.
   pdadmin> login
3. Enter the security master’s user name.
   Enter User ID: sec_master
4. Enter the security master’s password.
   Enter Password: password
   pdadmin>
5. Determine the name of your WebSEAL Server defined in Tivoli Access Manager. The name has following format: websealId-shortHostname. To list all the servers defined in Tivoli Access Manager, enter the following:
   pdadmin> server list
6. Create a WebSEAL junction.

The syntax of the command to create a WebSEAL junction is the following:

```
server task WebSEALServer create -t Type -h Hostname -p Portnumber -s -j -c ClientIdentityOptions /JunctionName
```

where `WebSEALServer` is the WebSEAL server name. This example uses `webseald-drbtest`.

- **-t** *Type* Junction type. Specify tcp.
- **-h** *Hostname* Fully-qualified hostname
- **-p** *Portnumber* Port number. The default value is 80 for a TCP junction.
- **-s** Stateful junction. This flag is required when Tivoli Identity Manager servers are replicated.
- **-j** Junction cookies. Cookies are used to handle server relative URLs.
- **-c** *ClientIdentityOptions* Choose a value that instructs WebSEAL to insert the "iv-user" HTTP header variable. For example, "iv_user". See the Tivoli Access Manager administration documentation for full options.
- **/JunctionName** A unique junction point name

For example, create a TCP junction by entering the following command on one line:

```
padmin> server task webseald-drbtest create -t tcp -h drbtest.tivoli.com -p 8080 -s -j -c iv_user /websphere
```

7. Create an access control list (ACL) requiring authenticated access to associate with the WebSEAL junction created. The syntax of the command to create an ACL is the following:

```
padmin> acl create aclName
```

For example:

```
padmin> acl create itim-acl
```

8. Add groups to the ACL using the following syntax:

```
padmin> acl modify aclName set group groupName permissions
```

For example:

```
padmin> acl modify itim-acl set group ITIM-Group Trx
padmin> acl modify itim-acl set unauthenticated T
padmin> acl modify itim-acl remove any-other
```

where the permissions are the following:

<table>
<thead>
<tr>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>traverse subdirectories</td>
</tr>
<tr>
<td>r</td>
<td>read</td>
</tr>
<tr>
<td>x</td>
<td>execute</td>
</tr>
</tbody>
</table>

9. Associate the ACL to the junction using the following syntax:

```
padmin> acl attach fullJunctionName aclName
```

For example:

```
padmin> acl attach /WebSEAL/drbtest/websphere/enrole itim-acl
```
10. Create an access control list allowing unauthenticated access. This ACL is associated with any object the end user can access without logging on. For example:
   `pdadmin> acl create unprotected-acl`

11. Add groups to the ACL requiring unauthenticated access. Use the following syntax:
   `pdadmin> acl attach fullJunctionName aclName`
   For example:
   `pdadmin> acl attach /WebSEAL/drptest/itim/enrole/self_reg unprotected-acl`

12. Update the Tivoli Identity Manager `ui.properties` file to configure one of the log off mechanisms provided in the `APP_WEB.war` directory. The logoff files provided are described in the table below.

   The `ssoLogout.jsp` and `websealLogout.jsp` files are only sample files that show the sample code required to use the Tivoli Identity Manager GUI logout button when WebSEAL single sign-on is enabled. You can edit these files (including language) to perform any functions appropriate to your environment.

<table>
<thead>
<tr>
<th>logoff.html</th>
<th>Default Tivoli Identity Manager logoff behavior:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(default)</td>
<td><strong>SSO disabled:</strong></td>
</tr>
<tr>
<td></td>
<td>• Upon logoff, presents the Tivoli Identity Manager logon page</td>
</tr>
<tr>
<td></td>
<td><strong>SSO enabled:</strong></td>
</tr>
<tr>
<td></td>
<td>• Upon logoff, returns you to the Tivoli Identity Manager GUI because the authentication information from Tivoli Access Manager (in the <code>iv-user</code> HTTP header) is still available.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ssoLogout.jsp</th>
<th>Use this sample file when you want the following combined behavior in a single sign-on environment:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Terminate the current Tivoli Identity Manager logon session and provide a link to return to the Tivoli Identity Manager GUI.</td>
</tr>
<tr>
<td></td>
<td>• Remain logged in to Tivoli Access Manager (<code>iv-user</code> HTTP header information is still available). This allows, for example, continued use of a portal page or to return to Tivoli Identity Manager without a logon prompt.</td>
</tr>
<tr>
<td></td>
<td>You can edit this file to customize the sample logoff functionality.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>websealLogout.jsp</th>
<th>Use this sample file when you want the following combined behavior in a single sign-on environment:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Terminate the Tivoli Identity Manager logon session.</td>
</tr>
<tr>
<td></td>
<td>• Terminate the Tivoli Access Manager logon session (<em>pkmslogout function is invoked</em>).</td>
</tr>
<tr>
<td></td>
<td><strong>pkmslogout</strong> only works for clients who use an authentication mechanism that does not supply authentication data with each request. For example, <strong>pkmslogout</strong> does not work for clients using Basic Authentication, certificates, or IP address information. In these cases, you must close the browser to log out. <strong>pkmslogout</strong> provides this information to the user in a message that appears on the logout page.</td>
</tr>
<tr>
<td></td>
<td>You can edit this file to customize the sample logoff functionality.</td>
</tr>
</tbody>
</table>

For example:
To create an SSL junction with Tivoli Identity Manager, do the following:

1. Start the iKeyman utility for the WebSphere Application Server.
2. Select Open in the Key Database File task.
3. Open the DummyServerKeyFile.jks file located in the WebSphere_root\etc directory. A password prompt is displayed. If you are using the dummy file, the password is "WebAS".
4. Select the websphere dummy server certificate and then click Extract Certificate.
5. On the Extract Certificate to a File dialog, enter the following:
   - Data type: Select Base64-encoded ASCII data.
   - Certificate file name: Enter the file name for the certificate.
   - Location: Enter the directory path where the certificate is to be stored. For this example, enter WebSphereServerCert.arm for the Certificate file name and store the certificate in the WebSphere_root\etc directory.
6. Click OK. Once the certificate is saved, the certificate needs to be transferred to the WebSEAL server.
   - If you defined your own keyfiles for WebSphere and obtained a certificate from a CA, you must use the root CA’s certificate which signed your WebSphere certificate in the following steps instead.
7. Close the WebSphere IBM Key Management GUI.
8. On the WebSEAL server, start the GSKit iKeyman executable.
9. Select Open in the Key Database File task.
10. This example uses the WebSEAL default database. Navigate to the WebSEAL_root\www-WebSEAL_instance\certs\pdsrv.kdb file and click Open.
11. Enter the password when a password prompt dialog appears. (The password for the default WebSEAL database is pdsrv.)
12. When the database opens, select Signer Certificates.
13. Click Add. The Add CA’s Certificate from a File dialog appears.
14. Do the following on the dialog to Add CA’s Certificate from a File:
   - Data type: Select Base64-encoded ASCII
   - Certificate file name: Click Browse to navigate to the certificate file name. This example uses the WebSphereServerCert.arm file located in the WebSphere_root\etc directory.
15. Click OK. A prompt appears for entry of a label name to store the certificate. This example uses the entry WAS 5 Server.
16. Click OK. The IBM Key Management panel appears with a list of Signer Certificates, including the label name that you specified.
17. Close the GSKit IBM Key Management GUI.
18. Type pdadmin at a command prompt to start the pdadmin command line interface.
19. Type `login` at the `pdadmin` command prompt and log in with the security master’s user name and password.

```bash
pdadmin> login
Enter User ID: sec_master
Enter Password: password
pdadmin>
```

20. Determine the name of your WebSEAL Server defined in Tivoli Access Manager. The name has following format: `webseald-shortHostname`. To list all the servers defined in Tivoli Access Manager, enter the following:

```bash
pdadmin> server list
```

21. Create a WebSEAL junction.

The syntax of the command to create a WebSEAL junction is the following:

```bash
server task WebSEALServer create -t Type -h Hostname -p Portnumber -s -j -c ClientIdentityOptions /JunctionName
```

- `Type`: Junction type. Specify `ssl`.
- `Hostname`: Fully-qualified hostname
- `Portnumber`: Port number. The default value is 9443 for an SSL junction.
- `Stateful`: Junction. This flag is required to improve performance when Tivoli Identity Manager servers are replicated.
- `Junction cookies`: Cookies are used to handle server relative URLs.
- `ClientIdentityOptions`: Choose a value that instructs WebSEAL to insert the "iv-user" HTTP header variable. For example, "iv_user". See the Tivoli Access Manager administration documentation for full options.
- `/JunctionName`: A unique junction point name

For example, create an SSL junction by entering the following command on one line:

```bash
pdadmin> server task webseald-drbtest create -t ssl -h drbtest.tivoli.com -p 9443 -s -j -c iv_user /websphere
```

22. Create an access control list (ACL) requiring authenticated access to associate with the WebSEAL junction created. The syntax of the command to create an ACL is the following:

```bash
pdadmin> acl create aclName
```

For example:

```bash
pdadmin> acl create itim-acl
```

23. Add groups to the ACL using the following syntax:

```bash
pdadmin> acl modify aclName set group groupName permissions
```

where permissions are the following:

<table>
<thead>
<tr>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>traverse subdirectories</td>
</tr>
<tr>
<td>r</td>
<td>read</td>
</tr>
<tr>
<td>x</td>
<td>execute</td>
</tr>
</tbody>
</table>

Table 2. Permissions
24. Associate the ACL to the junction using the following syntax:
   pdadmin> acl attach fullJunctionName aclName
   For example:
   pdadmin> acl attach /WebSEAL/drbtest/enrole itim-acl

25. Create an access control list requiring unauthenticated access. This ACL is
    associated with any object the end user can access without logging on.
    For example:
    pdadmin> acl create unprotected-acl

26. Add groups to the ACL requiring unauthenticated access. Use the following
    syntax:
    pdadmin> acl attach fullJunctionName aclName
    For example:
    pdadmin> acl attach /WebSEAL/drbtest/itim/enrole/self_reg unprotected-acl

27. Update the Tivoli Identity Manager ui.properties file to configure one of the
    log off mechanisms provided in the APP_WEB.war directory. The logoff files
    provided are described in the table below. The ssoLogout.jsp and websealLogout.jsp files are only sample files that
    show the sample code required to use the Tivoli Identity Manager GUI logout
    button when WebSEAL single sign-on is enabled. You can edit these files
    (including language) to perform any functions appropriate to your
    environment.

<table>
<thead>
<tr>
<th>logoff.html (default)</th>
<th>Default Tivoli Identity Manager logoff behavior:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSO disabled:</strong></td>
<td>• Upon logoff, presents the Tivoli Identity Manager logon page</td>
</tr>
<tr>
<td><strong>SSO enabled:</strong></td>
<td>• Upon logoff, returns you to the Tivoli Identity Manager GUI because the authentication information from Tivoli Access Manager (in the iv-user HTTP header) is still available.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ssoLogout.jsp</th>
<th>Use this sample file when you want the following combined behavior in a single sign-on environment:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Terminate the current Tivoli Identity Manager logon session and provide a link to return to the Tivoli Identity Manager GUI.</td>
<td></td>
</tr>
<tr>
<td>• Remain logged in to Tivoli Access Manager (iv-user HTTP header information is still available). This allows, for example, continued use of a portal page or to return to Tivoli Identity Manager without a logon prompt.</td>
<td></td>
</tr>
</tbody>
</table>

You can edit this file to customize the sample logoff functionality.
websealLogout.jsp

Use this sample file when you want the following combined behavior in a single sign-on environment:
- Terminate the Tivoli Identity Manager logon session.
- Terminate the Tivoli Access Manager logon session (**pkmslogout** function is invoked).

**pkmslogout** only works for clients who use an authentication mechanism that does not supply authentication data with each request. For example, **pkmslogout** does not work for clients using Basic Authentication, certificates, or IP address information. In these cases, you must close the browser to log out. **pkmslogout** provides this information to the user in a message that appears on the logout page.

You can edit this file to customize the sample logoff functionality.

For example:

```bash
enrole.ui.logoffURL=ssoLogout.jsp
```

28. Stop and restart the WebSphere Application Server to recognize the `ui.properties` file change.

### Specifying the URL for a Junction

The WebSEAL junction modifies the URL needed to access the Tivoli Identity Manager product. The syntax of the new URL is either of the following:

```
http://hostname/JunctionName/enrole/logon
https://hostname/JunctionName/enrole/logon
```

For example, use either of the following:

```
http://drbtest.tivoli.com/websphere/enrole/logon
https://drbtest.tivoli.com/websphere/enrole/logon
```
Part 3. Custom Report Configuration

Chapter 6. Installing and Using the Incremental Data Synchronizer ................................................................. 51
  Background Information for Incremental Data Synchronization ................................................................. 52
  The Role of the ACI changelog ......................................................................................................................... 52
  Enabling the Directory Server changelog ........................................................................................................ 53
  Incremental Data Synchronizer Installation Options ...................................................................................... 53
  Installing the Incremental Data Synchronizer on WebSphere/UNIX ............................................................ 53
    Separate Computer Installation .................................................................................................................... 54
    Same Computer Installation ....................................................................................................................... 55
  Installing the Incremental Data Synchronizer on WebSphere/Windows ...................................................... 56
    Separate Computer Installation .................................................................................................................. 56
    Same Computer Installation ..................................................................................................................... 57
  Installing the Incremental Data Synchronizer on WebLogic/UNIX ............................................................. 58
    Separate Computer Installation .................................................................................................................. 58
    Same Computer Installation ..................................................................................................................... 59
  Installing the Incremental Data Synchronizer on WebLogic/Windows ....................................................... 60
    Separate Computer Installation .................................................................................................................. 60
    Same Computer Installation ..................................................................................................................... 61
  Starting the Incremental Data Synchronizer .................................................................................................... 63
    Graphical Mode ......................................................................................................................................... 63
    Command Line Mode ............................................................................................................................... 65
  Fine-tuning the Incremental Data Synchronizer .............................................................................................. 66

Chapter 7. Configuring Crystal Reports ................................................................................................................ 67
  Process Flow When Using Crystal Reports ...................................................................................................... 68
  Configuring Tivoli Identity Manager to Use Crystal Reports ........................................................................... 69
    1. RAS Configuration (supported on Windows platform only) .................................................................... 69
    2a. Tivoli Identity Manager (WebSphere on Windows) Configuration ...................................................... 69
    2b. Tivoli Identity Manager (WebSphere on UNIX) Configuration .......................................................... 71
    2c. Tivoli Identity Manager Server (WebLogic on Windows) Configuration ............................................ 73
    2d. Tivoli Identity Manager (WebLogic on UNIX) Configuration .......................................................... 74
    3. Client Configuration (supported on Windows platform only) ............................................................... 75

Chapter 8. Designing Crystal Report Filter Conditions .......................................................................................... 77
  Tutorial on Designing Filter Conditions ......................................................................................................... 77
  Example Reports .............................................................................................................................................. 79
    Specifying JOIN conditions in Report Designer while designing reports ............................................... 79
    Account Attribute Report .......................................................................................................................... 79
    Person-Organization Roles Report ........................................................................................................... 80
    Person-Accounts Report ............................................................................................................................ 80

Chapter 9. Designing Custom Report Filter Conditions .......................................................................................... 83
  Tutorial on Designing Filter Conditions ......................................................................................................... 83
  Example Reports .............................................................................................................................................. 85
    Use of functions in reports .......................................................................................................................... 85
    Specifying JOIN conditions in Report Designer while designing reports ............................................... 85
    Account Attribute Report .......................................................................................................................... 85
    Person-Organization Roles Report ........................................................................................................... 86
    Person-Accounts Report ............................................................................................................................ 87
Chapter 6. Installing and Using the Incremental Data Synchronizer

The Incremental Data Synchronizer is a lightweight component that serves the Tivoli Identity Manager customized reporting process by providing a more optimal mechanism for synchronizing recently changed information. The Incremental Data Synchronizer provides real-time synchronization of data and ACI information between the Tivoli Identity Manager directory server and the Tivoli Identity Manager database. Additionally, the Incremental Data Synchronizer can be configured to enforce changes in schema entity or attribute mapping used in custom report templates.

The Incremental Data Synchronizer is a separate and optional component that can be configured after installing Tivoli Identity Manager. Incremental data synchronization is not a prerequisite for custom reports unless you require near real-time synchronization of data and ACIs.

For complete information on creating custom reports, please refer to the chapter entitled "Reports" in the IBM Tivoli Identity Manager Policy and Organization Administration Guide.

Section topics:
- “Background Information for Incremental Data Synchronization” on page 52
- “Enabling the Directory Server changelog” on page 53
- “Incremental Data Synchronizer Installation Options” on page 53
- “Installing the Incremental Data Synchronizer on WebSphere/UNIX” on page 53
- “Installing the Incremental Data Synchronizer on WebSphere/Windows” on page 56
- “Installing the Incremental Data Synchronizer on WebLogic/UNIX” on page 58
- “Installing the Incremental Data Synchronizer on WebLogic/Windows” on page 60
- “Starting the Incremental Data Synchronizer” on page 63
- “Fine-tuning the Incremental Data Synchronizer” on page 66
Background Information for Incremental Data Synchronization

Tivoli Identity Manager allows you to create new custom report templates using the resident Report Designer, or import report templates created using third-party tools (such as Crystal Reports). All custom reports are stored in the reporting tables within the Tivoli Identity Manager database. Access to the reporting system, including the ability to design and run reports, is protected through existing Tivoli Identity Manager ACI features.

Creating a custom report consists of the following steps:
1. Mapping attributes for reporting in the Schema Designer
2. Staging data
3. Designing report templates using the resident Report Designer, or importing report templates created using third-party tools
4. Granting report access to users

The Incremental Data Synchronizer synchronizes entire sets of staged entities and their ACI information. Additionally, it can enforce schema changes made through the Schema Designer. The Incremental Data Synchronizer performs the following operations:

1. Changelog synchronization:
   a. Fetch changelogs from the directory server
   b. Analyze effective operation and attribute values of each modified entry
   c. Update ACI information, if necessary
   d. Synchronize all available entry attributes in the database from analyzed changelogs

2. Schema enforcement:
   a. Determine changes made through the Schema Designer
   b. Map or unmap entities, if necessary
   c. Map or unmap entity attributes, if necessary
   d. Add or remove ACI information of newly mapped and unmapped attributes

The Incremental Data Synchronizer is a separate process that can be scheduled or invoked directly. Both changelog synchronization and schema enforcement features can be enabled and disabled independently as required.

The purpose of the Incremental Data Synchronizer is to ensure consistency of any changes between the Tivoli Identity Manager directory server and the staged reporting tables in the Tivoli Identity Manager database.

By allowing the Incremental Data Synchronizer to operate more frequently, changes in user account information are less likely to result in temporary lapses in the accuracy of information and proper interpretation of permissions for the custom report process. By reducing these lapses, the accuracy of the custom reporting process is enhanced.

The Role of the ACI changelog

The Incremental Data Synchronizer uses a mechanism known as the changelog, a feature provided by the directory server. The changelog is a history of changes
Enabling the Directory Server changelog

To enable the changelog for the specific directory server used by Tivoli Identity Manager, refer to the appropriate documentation provided by the vendor of that directory server.

Note: For IBM Directory Server, it is important to increase the default Application Heap Size of the DB2 database (used by the IBM Directory Server to store the changelog entries) to a large number such as 4096. Refer to the DB2 manuals for modifying the application heap size.

Incremental Data Synchronizer Installation Options

Select one of the installation sections below that is appropriate for your application server and platform:

- “Installing the Incremental Data Synchronizer on WebSphere/UNIX” on page 53
- “Installing the Incremental Data Synchronizer on WebSphere/Windows” on page 56
- “Installing the Incremental Data Synchronizer on WebLogic/UNIX” on page 58
- “Installing the Incremental Data Synchronizer on WebLogic/Windows” on page 60

Installing the Incremental Data Synchronizer on WebSphere/UNIX

The Incremental Data Synchronizer can be installed on the same computer as the WebSphere Application Server, or on a separate computer. It is recommended that you install the Incremental Data Synchronizer on a separate computer. Instructions for both types of installations are included in this section.

These procedures use the following conventions (reprinted from the Preface):

<table>
<thead>
<tr>
<th>Path Abbreviation</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITIM_HOME</td>
<td>INSTALL_DIR/ITIM/</td>
</tr>
<tr>
<td></td>
<td>INSTALL_DIR is the location of the Tivoli Identity Manager installation.</td>
</tr>
<tr>
<td>WAS_HOME</td>
<td>INSTALL_DIR/WebSphere/AppServer/</td>
</tr>
<tr>
<td></td>
<td>INSTALL_DIR is the location of the WebSphere installation.</td>
</tr>
<tr>
<td>WAS_NDM_HOME</td>
<td>INSTALL_DIR/WebSphere/DirectoryManager/</td>
</tr>
<tr>
<td></td>
<td>INSTALL_DIR is the location of the WebSphere installation.</td>
</tr>
<tr>
<td>BEA_HOME</td>
<td>INSTALL_DIR/bea/</td>
</tr>
<tr>
<td></td>
<td>INSTALL_DIR is the location of the WebLogic installation.</td>
</tr>
</tbody>
</table>
Separate Computer Installation

The following procedures describe setting up the Incremental Data Synchronizer on a separate computer from the WebSphere Application Server.

**Note:** `<synchronizer_computer>` is the computer where the Incremental Data Synchronizer is being installed. `<itim_computer>` is the computer where Tivoli Identity Manager has been installed.

1. Copy the `ITIM_HOME` directory from the `<itim_computer>` to the `<synchronizer_computer>`.
2. Copy the java directory from the `WAS_HOME` directory on the `<itim_computer>` to the `ITIM_HOME` directory on the `<synchronizer_computer>`.
3. Create a directory named `websphere_lib` in the `ITIM_HOME` directory on the `<synchronizer_computer>`.
4. Copy all the files and folders from `WAS_HOME/lib` directory on the `<itim_computer>` to the `ITIM_HOME/ websphere_lib` directory on the `<synchronizer_computer>`.
5. Copy the `app_ejb.jar`, `api_ejb.jar`, and `wf_ejb.jar` from the `WAS_HOME/installedApps/<computer_name>/enRole.ear` directory on the `<itim_computer>` to the `ITIM_HOME/lib` directory on the `<synchronizer_computer>`.
6. Copy the `implfactory.properties` file from the `WAS_HOME/properties` directory on the `<itim_computer>` to the `ITIM_HOME/data` directory on the `<synchronizer_computer>`.
7. Copy the `jaas_login_was.conf` file from the `ITIM_HOME/extensions/examples/apps/bin` directory on the `<itim_computer>` to the `ITIM_HOME/data` directory on the `<synchronizer_computer>`.
8. Create a directory named `logs` in `ITIM_HOME` on the `<synchronizer_computer>`.
9. Enable changelog processing in `adhocreporting.properties` on the `<itim_computer>` as follows:
   a. Set `changelogEnabled=true` in the `adhocreporting.properties` file in the data directory of the Tivoli Identity Manager Server installation.
   b. Set `changelogBaseDN=<changelog_base_dn>` in the `adhocreporting.properties` file in the data directory of the Tivoli Identity Manager Server installation. The `<changelog_base_dn>` is the base DN that holds the changelog entries in the directory server. For example:

```
changelogBaseDN=cn=changelog
```
10. Enable Schema enforcement in the `adhocreporting.properties` file on the `<itim_computer>` as follows:
    - Set `enableDeltaSchemaEnforcer=true` in the `adhocreporting.properties` file in the data directory of the Tivoli Identity Manager Server installation.
11. Modify the `enRole.properties` file in the `ITIM_HOME/data` directory on the `<synchronizer_computer>` as follows:
    
```
enrole.appServer.url=iiop://<itim_server_computer_name>:2809
```
12. Modify the `enRoleDatabase.properties` file in the `ITIM_HOME/data` directory on the `<synchronizer_computer>` to include the Tivoli Identity Manager database details.
13. Modify the enRoleLDAPConnection.properties file in the ITIM_HOME/data directory on the <synchronizer_computer> to include the Tivoli Identity Manager directory server details.

14. Modify the enRoleLogging.properties file in the ITIM_HOME/data directory on the <synchronizer_computer> as follows:

   log4j.appender.Logger.File=path_to_itim>.log

   The itim.log file path is ITIM_HOME/logs/itim.log. The logs directory was created in a previous step. For example:

   log4j.appender.Logger.File = ITIM_HOME/logs/itim.log

15. If the database used by Tivoli Identity Manager is DB2, copy the db2java.zip file from the SOLLIB/java12 directory on the <itim_computer> to the ITIM_HOME/lib directory on the <itimri_aci_synchronizer_machine>.

16. For the following two files, modify the ITIM_HOME variable to point to the ITIM_HOME directory:

   startIncrementalSynchronizerCMD_WAS.sh
   startIncrementalSynchronizerUI_WAS.sh

   For example:

   ITIM_HOME = ITIM_HOME

   These files can be found in the ITIM_HOME/bin/unix directory.

**Same Computer Installation**

The following procedures describe setting up the Incremental Data Synchronizer on the same computer as the WebSphere Application Server.

1. Copy the java directory from the WAS_HOME directory to the ITIM_HOME directory.

   **Note:** WAS_HOME is INSTALL_DIR/WebSphere/AppServer/

2. Create a directory named websphere_lib in the ITIM_HOME directory.

3. Copy all the files and folders from WAS_HOME/lib directory to the ITIM_HOME/websphere_lib directory.

4. Copy the files app_ejb.jar, api_ejb.jar, and wf_ejb.jar from WAS_HOME/installedApps/<computer_name>/enRole.ear to the ITIM_HOME/lib directory.

   The <computer_name> is usually the name of the computer on which the Tivoli Identity Manager Server is installed. This value is provided during the Tivoli Identity Manager installation. This name can be found in the directory path in the WebSphere installation directory where enRole.ear exists. For example:

   /usr/WebSphere/AppServer/installedApps/<computer_name>/enRole.ear

5. Copy the implfactory.properties file from WAS_HOME/properties to the ITIM_HOME/data directory.

6. Copy the jaas_login_was.conf file from the ITIM_HOME/extensions/examples/apps/bin directory to the ITIM_HOME/data directory.

7. Enable changelog processing in adhocreporting.properties as follows:

   a. Set changelogEnabled=true in the adhocreporting.properties file in the ITIM_HOME/data directory.

   b. Set changelogBaseDN=<changelog_base_dn> in the adhocreporting.properties file in the ITIM_HOME/data directory. The <changelog_base_dn> is the base DN that holds the changelog entries in the directory server. For example:

   changelogBaseDN=cn=changelog

8. Enable Schema enforcement in the adhocreporting.properties file as follows:
• Set enableDeltaSchemaEnforcer=true in the adhocreporting.properties file in the data directory of the Tivoli Identity Manager server.

9. If the database used by Tivoli Identity Manager is DB2, copy the db2java.zip file from the SQLLIB/java12 directory to the ITIM_HOME/lib directory.

Installing the Incremental Data Synchronizer on WebSphere/Windows

The Incremental Data Synchronizer can be installed on the same computer as the WebSphere Application Server, or on a separate computer. It is recommended that you install the Incremental Data Synchronizer on a separate computer. Instructions for both types of installations are included in this section.

Separate Computer Installation

The following procedures describe setting up the Incremental Data Synchronizer on a separate computer from the WebSphere Application Server.

Note: <synchronizer_computer> is the computer where the Incremental Data Synchronizer is being installed. <itim_computer> is the computer where Tivoli Identity Manager has been installed.

1. Copy the ITIM_HOME directory from the <itim_computer> to the <synchronizer_computer>.

2. Copy the java directory from the WAS_HOME directory on the <itim_computer> to the ITIM_HOME directory on the <synchronizer_computer>.

3. Create a directory named websphere_lib in the ITIM_HOME directory on the <synchronizer_computer>.

4. Copy all the files and folders from the WAS_HOME\lib directory on the <itim_computer> to the ITIM_HOME\websphere_lib directory on the <synchronizer_computer>.

5. Copy the files app_ejb.jar, api_ejb.jar, and wf_ejb.jar from the WAS_HOME\installedApps\<computer_name>\enRole.ear directory on the <itim_computer> to the ITIM_HOME\lib directory on the <synchronizer_computer>.

The <computer_name> is usually the name of the computer on which the Tivoli Identity Manager Server is installed. This value is provided during the Tivoli Identity Manager installation. This name can be found in the directory path in the WebSphere installation directory where enRole.ear exists. For example: C:\Program Files\WebSphere\AppServer\installedApps\<computer_name>\enRole.ear

6. Copy the implfactory.properties file from WAS_HOME\properties on the <itim_computer> to the ITIM_HOME\data directory on the <synchronizer_computer>.

7. Copy the jaas_login_was.conf file from the ITIM_HOME\extensions\examples\apps\bin directory on the <itim_computer> to the ITIM_HOME\data directory on the <synchronizer_computer>.

8. Create a directory named logs in ITIM_HOME on the <synchronizer_computer>.

9. Enable changelog processing in adhocreporting.properties on the <itim_computer> as follows:
   a. Set changelogEnabled=true in the adhocreporting.properties file in the data directory of the Tivoli Identity Manager Server installation.
   b. Set changelogBaseDN=<changelog_base_dn> in the adhocreporting.properties file in the data directory of the Tivoli Identity...
Manager Server installation. The <changelog_base_dn> is the base DN that holds the changelog entries in the directory server. For example:

c changelogBaseDN=cn=changelog

10. Enable Schema enforcement in the adhocreporting.properties file on the <itim_computer> as follows:

   • Set enableDeltaSchemaEnforcer=true in the adhocreporting.properties file in the data directory of the Tivoli Identity Manager Server installation.

11. Modify the enRole.properties file in the ITIM_HOME\data directory on the <synchronizer_computer> as follows:

   enrole.appServer.url=iiop://<itim_server_computer_name>:2809

12. Modify the enRoleDatabase.properties file in the ITIM_HOME\data directory on the <synchronizer_computer> to include the Tivoli Identity Manager database details.

13. Modify the enRoleLDAPConnection.properties file in the ITIM_HOME\data directory on the <synchronizer_computer> to include the Tivoli Identity Manager directory server details.

14. Modify the enRoleLogging.properties file in the ITIM_HOME\data directory on the <synchronizer_computer> as follows:

   log4j.appender.Logger.File=path_to_itim.log

   The itim.log file path is ITIM_HOME\logs\itim.log. The logs directory was created in a previous step. For example:

   log4j.appender.Logger.File=C:\\ITIM_HOME\logs\itim.log

15. If the database used by Tivoli Identity Manager is DB2, copy the db2java.zip file from the SQLLIB\java12 directory on the <itim_computer> to the ITIM_HOME\lib directory on the <itimri_aci_synchronizer_machine>.

16. For the following two files, modify the ITIM_HOME variable to point to the ITIM_HOME directory:

   startIncrementalSynchronizerCMD_WAS.bat
   startIncrementalSynchronizerUI_WAS.bat

   For example:

   set ITIM_HOME = C:\\ITIM_HOME

   These files can be found in the ITIM_HOME\bin\win directory.

**Same Computer Installation**

The following procedures describe setting up the Incremental Data Synchronizer on the same computer as the WebSphere Application Server.

1. Copy the java directory from the WAS_HOME directory to the ITIM_HOME directory.

2. Create a directory named websphere_lib in the ITIM_HOME directory.

3. Copy all the files and folders from WAS_HOME\lib directory to the ITIM_HOME\websphere_lib directory.

4. Copy the files app_ejb.jar, api_ejb.jar, and wf_ejb.jar from WAS_HOME\installedApps\<computer_name>\enRole.ear to the ITIM_HOME\lib directory.

   The <computer_name> is usually the name of the computer on which the Tivoli Identity Manager Server is installed. It is the value given during the Tivoli Identity Manager installation. This name can be found in the directory path in the WebSphere installation directory where enRole.ear exists. For example:

   C:\\Program Files\WebSphere\AppServer\installedApps\<computer_name>\enRole.ear
5. Copy the implFactory.properties file from {WAS_HOME}\properties to the {ITIM_HOME}\data directory.

6. Copy the jaas_login_was.conf file from the {ITIM_HOME}\extensions\examples\apps\bin directory to the {ITIM_HOME}\data directory.

7. Enable changelog processing in adhocreporting.properties as follows:
   a. Set changelogEnabled=true in the adhocreporting.properties file in the {ITIM_HOME}\data directory.
   b. Set changelogBaseDN=<changelog_base_dn> in the adhocreporting.properties file in the {ITIM_HOME}\data directory. The <changelog_base_dn> is the base DN that holds the changelog entries in the directory server. For example:
      changelogBaseDN=cn=changelog

8. Enable Schema enforcement in the adhocreporting.properties file as follows:
   • Set enableDeltaSchemaEnforcer=true in the adhocreporting.properties file in the data directory of the Tivoli Identity Manager Server installation.

9. If the database used by Tivoli Identity Manager is DB2, copy the db2java.zip file from the SQLIB\java12 directory to the {ITIM_HOME}\lib directory.

---

**Installing the Incremental Data Synchronizer on WebLogic/UNIX**

The Incremental Data Synchronizer can be installed on the same computer as the WebLogic Server, or on a separate computer. It is recommended that you install the Incremental Data Synchronizer on a separate computer. Instructions for both types of installations are included in this section.

---

**Separate Computer Installation**

The following procedures describe setting up the Incremental Data Synchronizer on a separate computer from the WebLogic Server.

**Note:** <synchronization_computer> is the computer where the Incremental Data Synchronizer is being installed. <itim_computer> is the computer where Tivoli Identity Manager has been installed.

1. Copy the {ITIM_HOME} directory from the <itim_computer> to the <synchronization_computer>.

2. Create a directory named java in the {ITIM_HOME} directory on the <synchronization_computer>.

3. Copy the contents of the jdk131_06 directory from the {BEA_HOME} directory on the <itim_computer> to the {ITIM_HOME}\java directory on the <synchronization_computer>.

4. Copy weblogic.jar from the {BEA_HOME}/weblogic700/server/lib directory on the <itim_computer> to the {ITIM_HOME}\lib directory on the <synchronization_computer>.

5. Copy jaas_login.conf from the {BEA_HOME}/user_projects/itim directory on the <itim_computer> to the {ITIM_HOME}\data directory on the <synchronization_computer>.

6. Create a directory named logs in the {ITIM_HOME} directory on the <synchronization_computer>.

7. Enable changelog processing in adhocreporting.properties on the <itim_computer> as follows:
a. Set changelogEnabled=true in the adhocreporting.properties file in the ITIM_HOME/data directory on the <itimputer>.

b. Set changelogBaseDN=<changelog_base_dn> in the adhocreporting.properties file in the ITIM_HOME/data directory on the <itimputer>. The <changelog_base_dn> is the base DN that holds the changelog entries in the directory server. For example:
   changelogBaseDN=cn=changelog

8. Enable Schema enforcement in the adhocreporting.properties file on the <itimputer> as follows:
   - Set enableDeltaSchemaEnforcer=true in the adhocreporting.properties file in the data directory of the Tivoli Identity Manager Server installation.

9. Modify the enRole.properties file in the ITIM_HOME/data directory on the <synchronizer_computer> as follows:
   enrole.appServer.url=t3://<itimputer>:<tim_server_port>
   <itimputer> is the name of the <itimputer>.
   <tim_server_port> is the port on which Tivoli Identity Manager is running.

**Note:** The port number is important. If not given, the application tries to connect to port 7001, which is the default port for WebLogic.

10. Modify the enRoleDatabase.properties file in the ITIM_HOME/data directory on the <synchronizer_computer> to include the Tivoli Identity Manager database details.

11. Modify the enRoleLDAPConnection.properties file in the ITIM_HOME/data directory on the <synchronizer_computer> to include the Tivoli Identity Manager directory server details.

12. Modify the enRoleLogging.properties file in the ITIM_HOME/data directory on the <synchronizer_computer> as follows:
   log4j.appender.Logger.File=path_to_itim.log
   The itim.log file path is ITIM_HOME/logs/itim.log. The logs directory was created in a previous step. For example:
   log4j.appender.Logger.File = ITIM_HOME/logs/itim.log

13. For the following two files, modify the ITIM_HOME variable to point to the ITIM_HOME directory:
   startIncrementalSynchronizerCMD_WLS.sh
   startIncrementalSynchronizerUI_WLS.sh
   For example:
   ITIM_HOME = ITIM_HOME
   These files can be found in the ITIM_HOME/bin/unix directory.

**Same Computer Installation**

The following procedures describe setting up the Incremental Data Synchronizer on the same computer as the WebLogic Server.

1. Create a directory named java in the ITIM_HOME/java directory.

2. Copy the contents of the jdk131_06 directory in the BEA_HOME directory to the ITIM_HOME/java directory.

3. Copy weblogic.jar from the BEA_HOME/weblogic700/server/lib directory to the ITIM_HOME/lib directory.

4. Copy jaas_login.conf from the BEA_HOME/user_projects/itim directory to the ITIM_HOME/data directory.

5. Enable changelog processing in adhocreporting.properties as follows:
a. Set changelogEnabled=true in the adhocreporting.properties file in the ITIM_HOME/data directory.

b. Set changelogBaseDN=<changelog_base_dn> in the adhocreporting.properties file in the ITIM_HOME/data directory. The <changelog_base_dn> is the base DN that holds the changelog entries in the directory server. For example:
   changelogBaseDN=cn=changelog

6. Enable Schema enforcement in the adhocreporting.properties file as follows:
   • Set enableDeltaSchemaEnforcer=true in the adhocreporting.properties file in the data directory of the Tivoli Identity Manager Server installation.

7. Modify the enRole.properties file in the ITIM_HOME/data directory as follows:
   enrole.appServer.url=t3://<itim_server_computer_name>:<itim_server_port>
   <itim_server_computer_name> is the name of the <itim_computer>. This is usually "localhost". <itim_server_port> is the port on which Tivoli Identity Manager is running.

   Note: The port number is important. If not given, the application tries to connect to port 7001, which is the default port for WebLogic.

---

Installing the Incremental Data Synchronizer on WebLogic/Windows

The Incremental Data Synchronizer can be installed on the same computer as the WebLogic Server, or on a separate computer. It is recommended that you install the Incremental Data Synchronizer on a separate computer. Instructions for both types of installations are included in this section.

Separate Computer Installation

The following procedures describe setting up the Incremental Data Synchronizer on a separate computer from the WebLogic Server.

Note: <synchronizer_computer> is the computer where the Incremental Data Synchronizer is being installed. <itim_computer> is the computer where Tivoli Identity Manager has been installed.

1. Copy the ITIM_HOME directory from the <itim_computer> to the <synchronizer_computer>.
2. Create a directory named java in the ITIM_HOME directory on the <synchronizer_computer>.
3. Copy the contents of the jdk131_06 directory in the BEA_HOME directory on the <itim_computer> to the ITIM_HOME\java directory on the <synchronizer_computer>.
4. Copy weblogic.jar from the BEA_HOME\weblogic700\server\lib directory on the <itim_computer> to the ITIM_HOME\lib directory on the <synchronizer_computer>.
5. Copy jaas_login.conf from the BEA_HOME\user_projects\itim directory on the <itim_computer> to the ITIM_HOME\data directory on the <synchronizer_computer>.
6. Create a directory named logs in the ITIM_HOME directory on the <synchronizer_computer>.
7. Enable changelog processing in adhocreporting.properties on the <itim_computer> as follows:
   a. Set changelogEnabled=true in the adhocreporting.properties file in the ITIM_HOME\data directory on the <itim_computer>.
b. Set changelogBaseDN=<changelog_base_dn> in the adhocreporting.properties file in the ITIM_HOME\data directory on the "itim_computer". The <changelog_base_dn> is the base DN that holds the changelog entries in the directory server. For example:
changelogBaseDN=cn=changelog

8. Enable Schema enforcement in the adhocreporting.properties file on the "itim_computer" as follows:
   - Set enableDeltaSchemaEnforcer=true in the adhocreporting.properties file in the data directory of the Tivoli Identity Manager Server installation.

9. Modify the enRole.properties file in the ITIM_HOME\data directory on the <synchronizer_computer> as follows:
enrole.appServer.url=t3://<tim_server_computer_name>:<tim_server_port>
<tim_server_computer_name> is the name of the "itim_computer".
<tim_server_port> is the port on which Tivoli Identity Manager is running.

Note: The port number is important. If not given, the application tries to connect to port 7001, which is the default port for WebLogic.

10. Modify the enRoleDatabase.properties file in the ITIM_HOME\data directory on the <synchronizer_computer> to include the Tivoli Identity Manager database details.

11. Modify the enRoleLDAPConnection.properties file in the ITIM_HOME\data directory on the <synchronizer_computer> to include the Tivoli Identity Manager directory server details.

12. Modify the enRoleLogging.properties file in the ITIM_HOME\data directory on the <synchronizer_computer> as follows:
log4j.appender.Logger.File=<path_to_itim>.log
The itim.log file path is ITIM_HOME\logs\itim.log. The logs directory was created in a previous step. For example:
log4j.appender.Logger.File=C:\ITIM_HOME\logs\itim.log

13. For the following two files, modify the ITIM_HOME variable to point to the ITIM_HOME directory:
    startIncrementalSynchronizerCMD_WLS.bat
    startIncrementalSynchronizerUI_WLS.bat
    For example:
    set ITIM_HOME = C:\ITIM_HOME
These files can be found in the ITIM_HOME\bin\win directory.

## Same Computer Installation
The following procedures describe setting up the Incremental Data Synchronizer on the same computer as the WebLogic Server.

1. Create a directory named java in the ITIM_HOME directory.
2. Copy the contents of the jdk131_06 directory in the BEA_HOME directory to the ITIM_HOME\java directory.
3. Copy weblogic.jar from the BEA_HOME\weblogic700\server\lib directory to the ITIM_HOME\lib directory.
4. Copy jaas_login.conf from the BEA_HOME\user_projects\itim directory to the ITIM_HOME\data directory.
5. Enable changelog processing in adhocreporting.properties as follows:
   a. Set changelogEnabled=true in the adhocreporting.properties file in the ITIM_HOME\data directory.
b. Set changelogBaseDN=<changelog_base_dn> in the adhocreporting.properties file in the ITIM_HOME\data directory. The <changelog_base_dn> is the base DN that holds the changelog entries in the directory server. For example:
changelogBaseDN=cn=changelog

6. Enable Schema enforcement in the adhocreporting.properties file as follows:
   • Set enableDeltaSchemaEnforcer=true in the adhocreporting.properties file in the data directory of the Tivoli Identity Manager Server installation.

7. Modify the enRole.properties file in the ITIM_HOME\data directory as follows:
enrole.appServer.url=t3://<tim_server_computer_name>:<tim_server_port>
<tim_server_computer_name> is the name of the <itim_computer>. This is usually "localhost". <tim_server_port> is the port on which Tivoli Identity Manager is running.

**Note:** The port number is important. If not given, the application tries to connect to port 7001, which is the default port for WebLogic.
Starting the Incremental Data Synchronizer

The Incremental Data Synchronizer can be invoked either in graphical user interface mode or in command line mode.

Note: The Tivoli Identity Manager Server needs to be started and data synchronized before starting the Incremental Data Synchronizer.

Graphical Mode

UNIX (WebSphere):

To invoke the Incremental Data Synchronizer in graphic user interface mode, run the following command located in the ITIM_HOME/bin/unix directory:

```
# startIncrementalSynchronizerUI_WAS.sh
```

Note: Use XClient to invoke the Incremental Data Synchronizer.

UNIX (WebLogic):

To invoke the Incremental Data Synchronizer in graphic user interface mode, run the following command located in the ITIM_HOME/bin/unix directory:

```
# startIncrementalSynchronizerUI_WLS.sh
```

Note: Use XClient to invoke the Incremental Data Synchronizer.

Windows (WebSphere):

To invoke the Incremental Data Synchronizer in graphic user interface mode, run the following command located in the ITIM_HOME/bin/win directory:

```
C:> startIncrementalSynchronizerUI_WAS.bat
```

Windows (WebLogic):

To invoke the Incremental Data Synchronizer in graphic user interface mode, run the following command located in the ITIM_HOME/bin/win directory:

```
C:> startIncrementalSynchronizerUI_WLS.bat
```

The following figure shows the Incremental Data Synchronizer when invoked in graphical mode:
Login:

To start ACI synchronization, the user must first provide the Tivoli Identity Manager Manager’s credentials. Click on the **Login** button to provide these credentials. The dialog box is shown in the following figure:

![Login Dialog Box](image)

*Figure 11. Incremental Data Synchronizer in graphical mode*

**Options:**

Now you must enter the base DN in the LDAP directory server, where the changelog entries are stored, and the time delay between two successive synchronizations.

**Note:** The time delay is the interval between the end of one synchronization and the start of the next synchronization.

To set these parameters click on the **Options** button. The dialog box is shown in the following figure:
Incremental Data Synchronizer operations:

Start the synchronization by clicking on the Start button. The progress of the synchronization and other details are shown in the text area of the graphical interface.

The synchronization process can be stopped by clicking on the Stop button.

The text area of the Incremental Data Synchronizer can be cleared by clicking on the Clear button.

The Exit button can be clicked to exit from the Incremental Data Synchronizer.

Command Line Mode

UNIX (WebSphere):

To invoke the Incremental Data Synchronizer in a command-line mode, run the following command located in the ITIM_HOME/bin/unix directory:

```
# startIncrementalSynchronizerCMD_WAS.sh itim-manager passwd chglog-base-dn time-int
```

UNIX (WebLogic):

To invoke the Incremental Data Synchronizer in a command-line mode, run the following command located in the ITIM_HOME/bin/unix directory:

```
# startIncrementalSynchronizerCMD_WLS.sh itim-manager passwd chglog-base-dn time-int
```

Windows (WebSphere):

To invoke the Incremental Data Synchronizer in a command-line mode, run the following command located in the ITIM_HOME\bin\win directory (entered as one line):

```
C:> startIncrementalSynchronizerCMD_WAS.bat itim-manager passwd chglog-base-dn time-int
```

Windows (WebLogic):

To invoke the Incremental Data Synchronizer in a command-line mode, run the following command located in the ITIM_HOME\bin\win directory (entered as one line):

```
C:> startIncrementalSynchronizerCMD_WLS.bat itim-manager passwd chglog-base-dn time-int
```
Where:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>itim-manager</td>
<td>Tivoli Identity Manager Manager’s login ID</td>
</tr>
<tr>
<td>passwd</td>
<td>Tivoli Identity Manager Manager’s password</td>
</tr>
<tr>
<td>chlog-base-dn</td>
<td>Base DN of Tivoli Identity Manager directory server changelog entries</td>
</tr>
<tr>
<td>time-int</td>
<td>Time interval between successive synchronizations in seconds</td>
</tr>
</tbody>
</table>

Example (UNIX/WebSphere):
```
# startIncrementalSynchronizerCMD_WAS.sh "itim manager" password cn=changelog 1800
```

Example (Windows/WebSphere) (entered as one line):
```
C:\> startIncrementalSynchronizerCMD_WAS.bat "itim manager" password cn=changelog 1800
```

**Note:** The time interval is the delay in seconds between the end of one synchronization and the start of the next synchronization.

---

**Fine-tuning the Incremental Data Synchronizer**

The Incremental Data Synchronizer can be fine-tuned by modifying properties located in the ad hoc reporting.properties configuration file. The following three properties can be modified in combination to produce efficient operation of the synchronization process:

- `changeLogFetchSize`
- `maximumChangeLogsToSynchronize`
- `changeLogsToAnalyzeBeforeSynchronization`

For detailed information on these properties and other configuration properties, refer to the ad hoc reporting.properties file reference in Chapter 11, “Configuring Supplemental Properties,” on page 127.
Chapter 7. Configuring Crystal Reports

Crystal Reports® version 9 is a report template creation and integration product from Crystal Decisions, Inc. You use the Crystal Reports designer tool to design sophisticated report templates. These templates can then be integrated into your Tivoli Identity Manager environment and made available to users who generate reports.

Crystal Reports functionality is an optional (not required) feature that allows you to provide user-defined custom reports.

You can integrate a custom Crystal Report template into the Tivoli Identity Manager environment by using the Tivoli Identity Manager GUI to import the report template file from the Crystal Reports client machine. The custom report template appears listed with the standard reports on the Reports panel of the Tivoli Identity Manager GUI. Users can now use this template to generate a custom report. Tivoli Identity Manager provides full ACI control over any report data displayed in a Crystal Report template.

Section topics:
- “Process Flow When Using Crystal Reports” on page 68
- “Configuring Tivoli Identity Manager to Use Crystal Reports” on page 69
Process Flow When Using Crystal Reports

Tivoli Identity Manager with the Crystal Reports functionality is a client/server system that is composed of the following components:

- **Crystal Reports client** (where Crystal Reports Designer tool is installed)
  Users can use the Crystal Reports Designer tool to design the report template.
- **Tivoli Identity Manager GUI**
  Administrators can make the new report template available to Tivoli Identity Manager users (using the "import" function).
- **Tivoli Identity Manager server**
  Sends a report request to the Crystal Reports Report Application Server (RAS)
- **Crystal Reports RAS (Report Application Server)**
  Processes the report by extracting data from the Tivoli Identity Manager database and populating the template with the data.
- **SDK** that provides an interface between the Tivoli Identity Manager server and the RAS

The following process flow explains the sequence of events when generating a report using a Crystal Reports template:

1. **On the Crystal Reports client machine**, use the Crystal Reports Designer tool to design a report template.
   Only those entities and columns that were mapped using the Tivoli Identity Manager Schema Designer are made available to the Crystal Reports Designer tool. After designing the report template, the Crystal Reports Designer tool can save the template as a report template file (.rpt) on the client file system.

2. **Using the Tivoli Identity Manager GUI**, make the new report template available to the Tivoli Identity Manager environment by "importing" the template (Import button in the Design Report task).

3. **When a user runs a report with this template selected**, the Tivoli Identity Manager server sends the report request to the RAS for processing.

4. **The RAS** uses the DSN (Data Source Name) created on the RAS machine to contact the Tivoli Identity Manager database, retrieve the appropriate data, and generate the complete report.
Configuring Tivoli Identity Manager to Use Crystal Reports

To import report templates designed using the Crystal Reports Designer Tool into the Tivoli Identity Manager environment, the Report Application Server (RAS) must be running on the network and the Tivoli Identity Manager server must be configured for the RAS SDK.

Configuration for Crystal Reports support in Tivoli Identity Manager involves three components:

1. Crystal Enterprise Report Application Server (RAS) configuration
   • Supported on Windows platform only
2. Tivoli Identity Manager application server configuration
   
   **Note:** Select one of the installation sections below that is appropriate for your application server and platform:
   a. WebSphere on Windows
   b. WebSphere on UNIX
   c. WebLogic on Windows
   d. WebLogic on UNIX
3. Client configuration (for the machine running the Crystal Reports Designer tool)
   • Supported on Windows platform only

1. RAS Configuration (supported on Windows platform only)

Perform the following steps to configure the Report Application Server (RAS):

1. Install the Crystal Reports Report Application Server (RAS) using the instructions provided with the software.
   The RAS can reside on the same Windows machine as the Tivoli Identity Manager application server or on a separate Windows machine.

2. Create a system DSN (Data Source Name) on the RAS to point to the Tivoli Identity Manager database used by the Tivoli Identity Manager server.

2a. Tivoli Identity Manager (WebSphere on Windows) Configuration

Perform the following steps to configure Tivoli Identity Manager with the WebSphere application server on a Windows machine:

1. From the RAS installation directory, copy all .jar files, except the 08405.jar/ebus405.jar and xerces.jar files, to the following directory on WebSphere:
   \WAS_HOME\installedApps\<machine-name>\enrole.ear\app_web\WEB-INF\lib\n
   **Note:** Create the lib folder if it does not exist.

   These .jar files are typically located in the following directory:
   C:\Program Files\Common Files\Crystal Decisions\2.0\jars

   **Note:** Please DO NOT overwrite the xerces.jar file that already exists in the target directory. Please DO NOT copy the 08405.jar or ebus405.jar file to the target directory. Either 08405.jar OR ebus405.jar file might be present in your Crystal installation folder, based on your build version of the Crystal Report Application Server.
2. Modify `ITIM_HOME\data\crystal.properties` file and update the following properties:
   - `crystalras`: Set this value to the server on which the RAS is installed
   - `dsn`: Set this value to the DSN created above
   - `database`: Set this value to the database being used by Tivoli Identity Manager

   For DB2 database configuration, if the database resides on the machine other than the RAS, specify the database alias that points to the actual database being used by Tivoli Identity Manager.

3. Restart the Tivoli Identity Manager server.

Create the `.war` file using the WebSphere Application Assembly Tool:

The following instructions allow you to deploy the `crystalreportviewers` directory as a `.war` file in WebSphere 5.0.

1. Start the WebSphere Application Assembly Tool:
   
   Start > Program Files > IBM WebSphere > Application Server 5.0 > Application Assembly Tool

2. Select **Web Module**.

3. Expand the node **Files**.

4. Right-click on **Resource Files** and select **Add Files**.

5. Select the `crystalreportviewers` directory from the Crystal installation directory. Typically this is found in:

   `C:\Program Files\Common Files\Crystal Decisions\2.0\crystalreportviewers`

6. Click **Add**.

   Confirm that the list of files in the `crystalreportviewers` directory appears in the bottom panel of the GUI.

7. Click **OK**.

8. Right click on **Jar Files** and select **Add Files**.

9. Select the `jars` directory from the Crystal installation directory. Typically this is found in:

   `C:\Program Files\Common Files\Crystal Decisions\2.0\jars`

10. Select all files with the `.jar` filename extension.

11. Click **Add**.

   Confirm that the list of files in the `jars` directory appears in the bottom panel of the GUI.

12. Click **OK**.

13. Click **Apply**.

14. Click on the node that is at the root of the tree in the left panel.

15. Edit the classpath to (entered as one line):

   `WEB-INF/lib/rascore.jar;
WEB-INF/lib/rasapp.jar;
WEB-INF/lib/webreporting.jar;
WEB-INF/lib/WebReportingWizard.jar;
WEB-INF/lib/Serialization.jar;
WEB-INF/lib/MetafileRenderer.jar;
WEB-INF/lib/ReportTemplate.jar;
WEB-INF/lib/CorbaIDL.jar;
WEB-INF/lib/OBBIDir.jar;
WEB-INF/lib/OBEvent.jar;
WEB-INF/lib/OBIMR.jar;
WEB-INF/lib/OBNaming.jar;`
WEB-INF/lib/OBProperty.jar;
WEB-INF/lib/OBTime.jar;
WEB-INF/lib/OBUtil.jar;
WEB-INF/lib/reportsourcefactory.jar

16. Click Apply.
17. Save the file as a .war file.

Deploy the .war file through the WebSphere Administration Console:
1. Start the WebSphere server.
2. Go to the WebSphere Administration Console:
   http://machine-name:9090/admin/
3. From the Administration Console, select:
   Applications > Install New Application
4. Select the local path if the file is being deployed from the machine other than
   server machine. Otherwise, select the full server path name.
5. Select the .war file created in the previous section.
6. Specify the context root as:
   /crystalreportviewers
7. Keep all default settings and specify the Application name as:
   crystalreportviewers
8. Click Finish.
9. Click Save to Master Configuration.
10. Click Save.
11. Expand the node Environment in the tree located in the left-hand frame.
13. Click OK in the right-hand frame.
14. Restart the WebSphere application server.

2b. Tivoli Identity Manager (WebSphere on UNIX)
Configuration

Perform the following steps to configure Tivoli Identity Manager with the
WebSphere application server on a UNIX machine:
1. From the RAS installation directory, copy all .jar files, except the
   08405.jar/ebus405.jar and xerces.jar files, to the following directory on
   WebSphere:
   WAS_HOME/installedApps/<machine-name>/enrole.ear/app_web/WEB-INF/lib/

   Note: Create the lib folder if it does not exist.

   These .jar files are typically located in the following directory:
   C:\Program Files\Common Files\Crystal Decisions\2.0\jars

   Note: Please DO NOT overwrite the xerces.jar file that already exists in the
   target directory. Please DO NOT copy the 08405.jar or ebus405.jar file
   to the target directory. Either 08405.jar OR ebus405.jar file might be
   present in your Crystal installation folder, based on your build version of
   the Crystal Report Application Server.
2. Modify ITIM_HOME/data/crystal.properties file and update the following
   properties:
   • crystalras: Set this value to the server on which the RAS is installed
- **dsn**: Set this value to the DSN created above.
- **database**: Set this value to the database being used by Tivoli Identity Manager.
  
  For DB2 database configuration, if the database resides on the machine other than the RAS, specify the database alias that points to the actual database being used by Tivoli Identity Manager.

3. Restart the Tivoli Identity Manager server.

**Create the .war file using the WebSphere Application Assembly Tool:**

The following instructions allow you to deploy the crystalreportviewers directory as a .war file in WebSphere 5.0.

1. Start the WebSphere Application Assembly Tool. The executable for this tool is located at:

   `WAS_HOME/bin/assembly.sh`

2. Select **Web Module**.

3. Expand the node **Files**.

4. Right-click on **Resource Files** and select **Add Files**.

5. Select the `crystalreportviewers` directory from the Crystal installation directory. Typically this is found in:

   `C:\Program Files\Common Files\Crystal Decisions\2.0\crystalreportviewers`

6. Click **Add**.

   Confirm that the list of files in the `crystalreportviewers` directory appears in the bottom panel of the GUI.

7. Click **OK**.

8. Right click on **Jar Files** and select **Add Files**.

9. Select the `jars` directory from the Crystal installation directory. Typically this is found in:

   `C:\Program Files\Common Files\Crystal Decisions\2.0\jars`

10. Select all files with the .jar filename extension.

11. Click **Add**.

   Confirm that the list of files in the `jars` directory appears in the bottom panel of the GUI.

12. Click **OK**.

13. Click **Apply**.

14. Click on the node that is at the root of the tree in the left panel.

15. Edit the classpath to (entered as one line):

   ```
   WEB-INF/lib/rascore.jar;
   WEB-INF/lib/rasapp.jar;
   WEB-INF/lib/webreporting.jar;
   WEB-INF/lib/WebReportingWizard.jar;
   WEB-INF/lib/Serialization.jar;
   WEB-INF/lib/MetafileRenderer.jar;
   WEB-INF/lib/ReportTemplate.jar;
   WEB-INF/lib/CorbaIDL.jar;
   WEB-INF/lib/OBBIDir.jar;
   WEB-INF/lib/OBEvent.jar;
   WEB-INF/lib/OBIMR.jar;
   WEB-INF/lib/OBNaming.jar;
   WEB-INF/lib/OBProperty.jar;
   WEB-INF/lib/OBTime.jar;
   WEB-INF/lib/OBUtil.jar;
   WEB-INF/lib/reportsourcefactory.jar
   ```
16. Click Apply.
17. Save the file as a .war file.

**Deploy the .war file through the WebSphere Administration Console:**
1. Start the WebSphere server.
2. Go to the WebSphere Administration Console:
   http://machine-name:9090/admin/
3. From the Administration Console, select:
   Applications > Install New Application
4. Select the local path if the file is being deployed from the machine other than server machine. Otherwise, select the full server path name.
5. Select the .war file created in the previous section.
6. Specify the context root as:
   /crystalreportviewers
7. Keep all default settings and specify the Application name as:
   crystalreportviewers
8. Click Finish.
9. Click Save to Master Configuration.
10. Click Save.
11. Expand the node Environment in the tree located in the left-hand frame.
13. Click OK in the right-hand frame.
14. Restart the WebSphere application server.

**2c. Tivoli Identity Manager Server (WebLogic on Windows) Configuration**

Perform the following steps to configure Tivoli Identity Manager with the WebLogic application server on a Windows machine:

1. From the RAS installation directory, copy all .jar files, except the 08405.jar/ebus405.jar and xerces.jar files, to the following directory on WebLogic:
   ITIM_HOME\lib
   These .jar files are typically located in the following directory:
   C:\Program Files\Common Files\Crystal Decisions\2.0\jars

   **Note:** Please DO NOT overwrite the xerces.jar file that already exists in the target directory. Please DO NOT copy the 08405.jar file to the target directory. Either 08405.jar OR ebus405.jar file might be present in your Crystal installation folder, based on your build version of the Crystal Report Application Server.

2. Modify ITIM_HOME\data\crystal.properties file and update the following properties:
   - **crystalras**: Set this value to the server on which the RAS is installed
   - **dsn**: Set this value to the DSN created above
   - **database**: Set this value to the database being used by Tivoli Identity Manager

3. Restart the Tivoli Identity Manager server.

**WebLogic configuration:**
The following instructions allow you to deploy the `crystalreportviewers` directory in WebLogic 7.0.

1. Create the `DefaultWebapp_myserver` sub-directory in
   `BEA_HOME/user_projects/itim/applications`
2. Copy `crystalreportviewers` from the Crystal installation directory to this
directory. Typically the Crystal installation directory is:
   `C:\Program Files\Common Files\Crystal Decisions\2.0`
3. Start the WebLogic server.
4. Using the browser, access the WebLogic console:
   `http://machine-name/console`
5. Login using the "system" user and the password provided. The default
   password is "enrole".
6. Expand the tree on the left side of screen:
   `itim > Deployments > Web Application > DefaultWebapp_myserver`
7. Click on Targets tab.
8. If `myserver` exists in the Available list, click on the arrow button to add it to the
   Chosen list.
9. Click Apply.

2d. Tivoli Identity Manager (WebLogic on UNIX) Configuration

Perform the following steps to configure Tivoli Identity Manager with the
WebLogic application server on a UNIX machine:

1. From the RAS installation directory, copy all .jar files, except the
   `08405.jar/ebus405.jar` and `xerces.jar` files, to the following directory on
   WebLogic:
   `ITIM_HOME/lib`
   These .jar files are typically located in the following directory:
   `C:\Program Files\Common Files\Crystal Decisions\2.0\jars`
   
   **Note:** Please DO NOT overwrite the `xerces.jar` file that already exists in the
   target directory. Please DO NOT copy the `08405.jar` or `ebus405.jar` file
   to the target directory. Either `08405.jar` OR `ebus405.jar` file might be
   present in your Crystal installation folder, based on your build version of
   the Crystal Report Application Server.

2. Modify `ITIM_HOME/data/crystal.properties` file and update the following
   properties:
   - `crystalras`: Set this value to the server on which the RAS is installed
   - `dsn`: Set this value to the DSN created above
   - `database`: Set this value to the database being used by Tivoli Identity
     Manager

3. Restart the Tivoli Identity Manager server.

WebLogic configuration:

The following instructions allow you to deploy the `crystalreportviewers` directory in WebLogic 7.0.

1. Create the `DefaultWebapp_myserver` sub-directory in
   `BEA_HOME/user_projects/itim/applications`
2. Copy `crystalreportviewers` from the Crystal installation directory to this
directory. Typically the Crystal installation directory is:
C:\Program Files\Common Files\Crystal Decisions\2.0

3. Start the WebLogic server.

4. Using the browser, access the WebLogic console:
   http://machine-name/console

5. Login using the "system" user and the password provided. The default password is "enrole".

6. Expand the tree on the left side of screen:
   im > Deployments > Web Application > DefaultWebapp_myslap

7. Click on Targets tab.

8. If myserver exists in the Available list, click on the arrow button to add it to the Chosen list.

9. Click Apply.

3. Client Configuration (supported on Windows platform only)

The following configuration is required for the client machine running the Crystal Reports 9 Designer tool software.

- Create a system DSN (Data Source Name) on the client machine running Crystal Reports 9 Designer tool that points to the Tivoli Identity Manager database used by the Tivoli Identity Manager server.
  This connection is important when the Designer tool retrieves the appropriate entities and column information from the Tivoli Identity Manager database.
- Use the "enrole" user name with the appropriate password. This user should have access to the database tables on the Tivoli Identity Manager server. The password can be obtained from the Tivoli Identity Manager administrator.

Note: Do not use the tables starting with "user_" when designing a Crystal Report.

Note: The SUPERVISOR and CONTAINERDN columns of the staged tables contain metadata information and should not be used in the designed Crystal Report.
Chapter 8. Designing Crystal Report Filter Conditions

Section topics:
- “Tutorial on Designing Filter Conditions” on page 77
- “Example Reports” on page 79

Tutorial on Designing Filter Conditions

The filter conditions have a very close mapping to the SQL language that is used for accessing databases. The filter conditions designed using Crystal Report Designer will be translated to SQL and executed during report generation.

The Crystal Report designer has to first select the tables that will be used in the report in “Database Expert”. The user sees all the tables created during synchronization. The following things should be noted while selecting the tables:

- The names of the tables that are generated for multi-valued attributes are:
  `<ENTITY_NAME>_<ATTRIBUTE_NAME>`
  However, the `<ATTRIBUTE_NAME>` used here is not the display name in Tivoli Identity Manager, but the internal representation of this attribute. (For example `cn` for Full Name).

- The scheme that is used for generating table names during data synchronization is:
  `<ENTITY_NAME>_<ATTRIBUTE_NAME>`
  However, Oracle database does not support an identifier name of more than 30 characters. Hence the tables created during the Data Synchronization cannot have names that exceeded 30 characters.
  In cases where this name extends beyond 30 characters, the name of the table would be generated using the following scheme:
  First 22 characters of the name ( `<ENTITY_NAME>_<ATTRIBUTE_NAME>` ) + `<unique-ID>`
  This scheme ensures that the names of the tables can never exceed 30 characters.
  The Report Designer would have to find out the multi-valued attribute (for which the table is created) in some cases by looking at the columns of the tables since the table names may be cryptic.

- User should not select any of the USER_<ENTITY> tables in Database Expert. These tables have ACI information and the data is not meant for reporting purposes.

Given below is a list of entities, their attributes and their relationships with each other in the Tivoli Identity Manager context. This list can prove to be very useful for creating the right filter conditions while designing reports. The Filters column in the table below also gives the exact JOIN condition that can be used in the report for accurate and meaningful results.
<table>
<thead>
<tr>
<th>Serial No.</th>
<th>Entities</th>
<th>Filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Person, Account</td>
<td>Person.DN = Account.owner</td>
</tr>
<tr>
<td>2</td>
<td>Person_erroles, DefaultRole</td>
<td>Person_erroles.erroles= DefaultRole.DN</td>
</tr>
<tr>
<td>3</td>
<td>Person, OrganizationalUnit</td>
<td>Person.ParentDN = OrganizationalUnit.DN OR OrganizationalUnit.ersupervisor = Person.DN</td>
</tr>
<tr>
<td>4</td>
<td>Account, Service</td>
<td>Account.erservice = Service.DN</td>
</tr>
<tr>
<td>5</td>
<td>Provisioning Policy, Organization Role</td>
<td>ProvisioningPolicy.erparent = OrganizationUnit.DN</td>
</tr>
<tr>
<td>6</td>
<td>Location, Person</td>
<td>Location.ersupervisor = Person.DN</td>
</tr>
<tr>
<td>7</td>
<td>BPOrganization, Person</td>
<td>BPOrganization.ersponsor = Person.DN</td>
</tr>
<tr>
<td>8</td>
<td>BP Person, DefaultRole</td>
<td>BPBPerson.erroles = DefaultRole.DN</td>
</tr>
<tr>
<td>9</td>
<td>Organization, Location</td>
<td>Organization.DN = Location.erpartment</td>
</tr>
<tr>
<td>10</td>
<td>Organization, OrganizationalUnit</td>
<td>Organization.DN = OrganizationUnit.erpartment</td>
</tr>
<tr>
<td>11</td>
<td>Organization, BPOrganization</td>
<td>Organization.DN = BPOrganization.erpartment</td>
</tr>
<tr>
<td>12</td>
<td>OrganizationalUnit, Location</td>
<td>OrganizationalUnit.erpartment = Location.DN OR Location.erpartment = OrganizationalUnit.DN</td>
</tr>
<tr>
<td>13</td>
<td>OrganizationalUnit, BPOrganization</td>
<td>OrganizationalUnit.erpartment = BPOrganization.DN OR BPOrganization.erpartment = OrganizationalUnit.DN</td>
</tr>
<tr>
<td>14</td>
<td>Location, BPOrganization</td>
<td>Location.erpartment = BPOrganization.DN OR BPOrganization.erpartment = Location.DN</td>
</tr>
<tr>
<td>15</td>
<td>Service, Person</td>
<td>Service.owner = Person.DN</td>
</tr>
<tr>
<td>16</td>
<td>SQL2000Account, Service</td>
<td>SQL2000Account.erservice = Service.DN</td>
</tr>
<tr>
<td>17</td>
<td>ITIMAccount, ITIM Service</td>
<td>ITIMAccount.erservice = ITIMService.DN</td>
</tr>
<tr>
<td>18</td>
<td>Entitlement, Service</td>
<td>Service.DN = Entitlement.ServiceTargetName</td>
</tr>
<tr>
<td>19</td>
<td>ProvisioningPolicy, Entitlement</td>
<td>ProvisioningPolicy.DN = Entitlement.DN</td>
</tr>
<tr>
<td>20</td>
<td>ACI, ACI_Principals</td>
<td>ACI.DN = ACI_Principals.DN AND ACI.Name = ACI_Principals.Name AND ACI.Target = ACI_Principals.Target</td>
</tr>
<tr>
<td>21</td>
<td>ACI, ACI_Permission_ClassRight</td>
<td>ACI.DN = ACI_Permission_ClassRight.DN AND ACI.Name = ACI_Permission_ClassRight.Name AND ACI.Target = ACI_Permission_ClassRight.Target</td>
</tr>
<tr>
<td>22</td>
<td>ACI, ACI_Permission_AttributeRight</td>
<td>ACI.DN = ACI_Permission_AttributeRight.DN AND ACI.Name = ACI_Permission_AttributeRight.Name AND ACI.Target = ACI_Permission_AttributeRight.Target</td>
</tr>
</tbody>
</table>
For Reference:

<table>
<thead>
<tr>
<th>Display Name in ITIM</th>
<th>Internal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organizational Role</td>
<td>DefaultRole</td>
</tr>
<tr>
<td>Business Partner Organization</td>
<td>BPOrganization</td>
</tr>
<tr>
<td>Business partner Person</td>
<td>BPPerson</td>
</tr>
</tbody>
</table>

Example Reports

Described below are some mechanisms for designing filter conditions. These explain the process of designing filters by taking a few examples and explaining them using some sample data. They also reflect some relationships between different columns of some synchronized tables.

**Specifying JOIN conditions in Report Designer while designing reports**

The user must specify appropriate filters while designing reports in report designer so that the desired data is fetched when the report is run.

Here are a few examples:

**Account Attribute Report**

(All Users with accounts of type ITIMService)

Design the report as:

- **Report columns**: Account.eruid, ITIM.Service.erservicename
- **Filters**: None

Hence, if there are two users:

- User1 with ITIMService1 and ITIMService2
- User2 with ITIMService3

The result would be:

<table>
<thead>
<tr>
<th>User</th>
<th>ITIMService</th>
</tr>
</thead>
<tbody>
<tr>
<td>User1</td>
<td>ITIMService1</td>
</tr>
<tr>
<td>User1</td>
<td>ITIMService2</td>
</tr>
<tr>
<td>User1</td>
<td>ITIMService3</td>
</tr>
<tr>
<td>User2</td>
<td>ITIMService1</td>
</tr>
<tr>
<td>User2</td>
<td>ITIMService2</td>
</tr>
<tr>
<td>User2</td>
<td>ITIMService3</td>
</tr>
</tbody>
</table>

This is nothing but the Cartesian product of the two tables.

To get the proper result set, appropriate JOIN conditions specifying the relationships between these two tables will have to be specified.
The JOIN condition in this case should be:

- **Filters**: Account.Service = ITIMService.DN

With this filter the result would be:

<table>
<thead>
<tr>
<th>User</th>
<th>ITIMService</th>
</tr>
</thead>
<tbody>
<tr>
<td>User1</td>
<td>ITIMService1</td>
</tr>
<tr>
<td>User1</td>
<td>ITIMService2</td>
</tr>
<tr>
<td>User2</td>
<td>ITIMService3</td>
</tr>
</tbody>
</table>

**Person-Organization Roles Report**
(For persons associated with an Organization Role)

Design the report as:

- **Report columns**: Person.cn, DefaultRole.Name
- **Filter**: DefaultRole.Name = '_USERINPUT_'

Hence, if there are two users:

- Person1 with Role1
- Person2 with Role2

If "Role1" is given as input during report generation, the result would be:

<table>
<thead>
<tr>
<th>Person</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person1</td>
<td>Role1</td>
</tr>
<tr>
<td>Person2</td>
<td>Role1</td>
</tr>
</tbody>
</table>

For this report to generate the correct results, the following filter conditions are required:

Person.erroles = DefaultRole.DN
AND DefaultRole.Name = '_USERINPUT_'

If "Role1" is given as input during report generation, the result would be:

<table>
<thead>
<tr>
<th>Person</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person1</td>
<td>Role1</td>
</tr>
</tbody>
</table>

The specified filter condition works because the “Organization Roles” attribute of the "Person" entity contains the value of the DN of the role that the user belongs to. Also note that Organization Role is a multi-valued attribute, that is, a person can have multiple Roles in an Organization.

**Person-Accounts Report**
(Accounts associated with persons in the system)

Design the report as:

- **Report column**: Person.cn, Account.eraccountstatus
- **Filters**: None

This report would return the Cartesian product of the Person and account table entries.

The JOIN condition that specifies the relationship between Person and Account tables is:
Account.owner = Person.DN
Chapter 9. Designing Custom Report Filter Conditions

Section topics:
• “Tutorial on Designing Filter Conditions” on page 83
• “Example Reports” on page 85

Tutorial on Designing Filter Conditions

The filter conditions have a very close mapping to the SQL language that is used for accessing databases. The filter conditions designed using Report Designer will be translated to SQL and executed during report generation.

Given below is a list of entities, their attributes and their relationships with each other in the ITIM context. This list can prove to be very useful for creating the right filter conditions while designing reports. The Filters column in the table below gives the exact JOIN condition that can be used in the report for accurate and meaningful results.

The tutorial concludes with some example reports that explain how and when these JOIN conditions can be used while designing reports.

<table>
<thead>
<tr>
<th>Serial No.</th>
<th>Entities</th>
<th>Filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Person, Account</td>
<td>Person.DN = Account.owner</td>
</tr>
<tr>
<td>2</td>
<td>Person, Organization Role</td>
<td>Person.Organization Roles = Organization Role.DN</td>
</tr>
<tr>
<td>3</td>
<td>Person, Organizational Unit</td>
<td>Person.ParentDN = Organizational Unit.DN OR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Organizational Unit. Supervisor = Person.DN</td>
</tr>
<tr>
<td>4</td>
<td>Account, Service</td>
<td>Account.Service = Service.DN</td>
</tr>
<tr>
<td>5</td>
<td>Provisioning Policy, Organization Role</td>
<td>Organization Role.DN = getDN(Provisioning Policy.Policy Membership) **</td>
</tr>
<tr>
<td>6</td>
<td>Provisioning Policy, Organizational Unit</td>
<td>Provisioning Policy.Parent DN = Organizational Unit.DN</td>
</tr>
<tr>
<td>7</td>
<td>Provisioning Policy, Service</td>
<td>Service.DN = getDN(Provisioning Policy.Policy Target ) **</td>
</tr>
<tr>
<td>8</td>
<td>Location, Person</td>
<td>Location.Supervisor = Person.DN</td>
</tr>
<tr>
<td>9</td>
<td>Business Partner Organization, Person</td>
<td>Business Partner Organization.Sponsor = Person.DN</td>
</tr>
<tr>
<td>10</td>
<td>Business Partner Person, Organization Role</td>
<td>Business Partner Person.Organization Roles = Organization Role.DN</td>
</tr>
<tr>
<td>11</td>
<td>Organization, Location</td>
<td>Organization.DN = Location.Parent DN</td>
</tr>
</tbody>
</table>
|   | **Organization, Organizational Unit, Business Partner Organization** | **Organization.DN = Organizational Unit.ParentDN**  
|   |                       | **Organization.DN = Business Partner**  
|   |                       | **Organization.ParentDN**  
| 12 | **Organization, Organizational Unit, Location** | **Organizational Unit.Parent DN = Location.DN**  
|   |                       | **Location.Parent DN = Organizational Unit.DN**  
| 13 | **Organizational Unit, Business Partner Organization** | **Organizational Unit.Parent DN = Business Partner Organization DN**  
|   |                       | **Business Partner Organization.Parent DN = Organizational Unit.DN**  
| 14 | **Location, Business Partner Organization** | **Location..Parent DN = Business Partner Organization.DN**  
|   |                       | **Business Partner Organization.Parent DN = Location.DN**  
| 15 | **Service, Person** | **Service.Account Owner = Person.DN**  
| 16 | **SQL.2000Account, Service** | **SQL2000Account.Service = Service.DN**  
| 17 | **ITIMAccount, ITIM Service** | **ITIMAccount.Service = ITIMService.DN**  
| 18 | **Entitlement, Service** | **Service.DN = Entitlement.Service Target Name**  
| 19 | **ProvisioningPolicy, Entitlement** | **ProvisioningPolicy.DN = Entitlement.DN**  
| 20 | **ACI, ACI Principals** | **ACI.DN = ACI Principals.DN**  
|   |                       | **AND ACI.Name = ACI Principals.Name**  
|   |                       | **AND ACI.Target = ACI Principals.Target**  
| 21 | **ACI, ACI Permission ClassRight** | **ACI.DN = ACI Permission ClassRight.DN**  
|   |                       | **AND ACI.Name = ACI Permission ClassRight.Name**  
|   |                       | **AND ACI.Target = ACI Permission ClassRight.Target**  
| 22 | **ACI, ACI Permission AttributeRight** | **ACI.DN = ACI Permission AttributeRight.DN**  
|   |                       | **AND ACI.Name = ACI Permission AttributeRight.Name**  
|   |                       | **AND ACI.Target = ACI Permission AttributeRight.Target**  
| 23 | **ACI, ACI Role DNs** | **ACI.DN = ACI Role DNs.DN**  
|   |                       | **AND ACI.Name = ACI Role DNs.Name**  
|   |                       | **AND ACI.Target = ACI Role DNs.Target**  
| 24 | **ACI, Organizational Unit** | **ACI.DN = Organizational Unit.DN**  
| 25 |

**Note:**

getDN is a function provided by the Report Designer to extract DN from the columns where data is in the format:

```<number>;<dn>```

For example, the column ProvisioningPolicy.Policymembership has entries of the format:

```<number>;<dn>```
In order to extract the DN out of this string, the \textbf{getDN} function can be used. An example of such filter can be:

\texttt{Organization Role.DN = getDN(ProvisioningPolicy.Policymembership)}

**Example Reports**

Described below are some mechanisms for designing filter conditions. These explain the process of designing filters by taking a few examples and explaining them using some sample data. They also reflect some relationships between different columns of some synchronized tables.

**Use of functions in reports**

You can use functions with the various columns in report design. The default functions provided with the implementation are:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper</td>
<td>This function takes one argument and transforms the argument to uppercase.</td>
</tr>
<tr>
<td>Lower</td>
<td>This function takes one argument and transforms the argument to lowercase.</td>
</tr>
<tr>
<td>GetDN</td>
<td>This is primarily meant to extract the DN from columns which contain a string of the format: \texttt{&lt;number&gt;;&lt;dn&gt;}</td>
</tr>
</tbody>
</table>

For example:

The column \texttt{ProvisioningPolicy.Policymembership} has the entries of the format:

\texttt{<number>;<dn>}

In order to extract the DN out of this string \texttt{getDN} function can be used. An example of such filter can be:

\texttt{Organization Role.DN = getDN(ProvisioningPolicy.Policymembership)}

**Specifying JOIN conditions in Report Designer while designing reports**

The user must specify appropriate filters while designing reports in report designer so that the desired data is fetched when the report is run.

Here are a few examples:

**Account Attribute Report**

(All Users with accounts of type ITIMService)

Design the report as:

- \textbf{Report columns}: Account.Userid, ITIM.ServiceName
- \textbf{Filters}: None

Hence, if there are two users:

- User1 with ITIMService1 and ITIMService2
- User2 with ITIMService3
The result would be:

<table>
<thead>
<tr>
<th>User</th>
<th>ITIMService</th>
<th>User</th>
<th>ITIMService</th>
<th>User</th>
<th>ITIMService</th>
</tr>
</thead>
<tbody>
<tr>
<td>User1</td>
<td>ITIMService1</td>
<td>User1</td>
<td>ITIMService2</td>
<td>User1</td>
<td>ITIMService3</td>
</tr>
<tr>
<td>User2</td>
<td>ITIMService1</td>
<td>User2</td>
<td>ITIMService2</td>
<td>User2</td>
<td>ITIMService3</td>
</tr>
</tbody>
</table>

This is nothing but the Cartesian product of the two tables.

To get the proper result set, appropriate JOIN conditions specifying the relationships between these two tables will have to be specified.

The JOIN condition in this case should be:

- **Filters**: Account.Service = ITIM.DN

With this filter the result would be:

<table>
<thead>
<tr>
<th>User</th>
<th>ITIMService</th>
<th>User</th>
<th>ITIMService</th>
</tr>
</thead>
<tbody>
<tr>
<td>User1</td>
<td>ITIMService1</td>
<td>User1</td>
<td>ITIMService2</td>
</tr>
<tr>
<td>User2</td>
<td>ITIMService1</td>
<td>User2</td>
<td>ITIMService3</td>
</tr>
</tbody>
</table>

**Person-Organization Roles Report**
(For persons associated with an Organization Role)

Design the report as:

- **Report columns**: Person.FullName, OrganizationRole.Name
- **Filter**: OrganizationRole.Name = '_USERINPUT_'

Hence, if there are two users:

- Person1 with Role1
- Person2 with Role2

If "Role1" is given as input during report generation, the result would be:

<table>
<thead>
<tr>
<th>Person</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person1</td>
<td>Role1</td>
</tr>
<tr>
<td>Person2</td>
<td>Role1</td>
</tr>
</tbody>
</table>

For this report to generate the correct results, the following filter conditions are required:

- Person.OrganizationRoles = OrganizationRole.DN
- AND OrganizationRole.Name = '_USERINPUT_'

If "Role1" is given as input during report generation, the result would be:

<table>
<thead>
<tr>
<th>Person</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person1</td>
<td>Role1</td>
</tr>
</tbody>
</table>
The specified filter condition works because the "Organization Roles" attribute of the "Person" entity contains the value of the DN of the role that the user belongs to. Also note that Organization Role is a multi-valued attribute, that is, a person can have multiple Roles in an Organization.

**Person-Accounts Report**
(Accounts associated with persons in the system)

Design the report as:
- **Report column**: Person.FullName, Account.AccountStatus
- **Filters**: None

This report would return the Cartesian product of the Person and account table entries.

The JOIN condition that specifies the relationship between Person and Account tables is:

```
Account.owner = Person.DN
```
### Part 4. Property File Reference

#### Chapter 10. Configuring System Properties
- Understanding Properties Files ........................................... 91
- WebLogic-specific Configuration .......................................... 93
- WebSphere-specific Configuration ......................................... 97
- Application Server Information .......................................... 101
- Default Tenant Information ............................................... 103
- LDAP Server Information ................................................. 104
- Cache Information ................................................................ 105
- Messaging Information ...................................................... 106
- Scheduling Information ..................................................... 109
- Password Transaction Monitor Settings ................................ 110
- XML and DTD Information .................................................. 111
- LDAP Connection Pool Information ..................................... 112
- System Configuration Program .......................................... 114
- Workflow Configuration Information .................................... 115
- Mail Services Configuration ............................................... 119
- Reconciliation Information ................................................. 120
- Shared Secret Hashing ....................................................... 122
- SSL Two-way Authentication Properties .............................. 123
- Request Management UI Configuration ............................... 124
- Application Client Request Configuration ............................ 125

#### Chapter 11. Configuring Supplemental Properties
- Understanding Properties Files .......................................... 127
- adhocreporting.properties .................................................. 128
- crystal.properties ................................................................ 129
- DataBaseFunctions.conf ..................................................... 133
- enRoleAuthentication.properties ......................................... 134
  - Configuring a custom authentication mechanism ................ 135
- enRoleDatabase.properties ................................................. 137
- enRoleLDAPConnection.properties ..................................... 141
- enRoleLogging.properties .................................................. 144
- enRoleMail.properties ...................................................... 147
- enrolepolicies.properties ................................................... 149
- enroleworkflow.properties ............................................... 151
- fesicontections.properties ............................................... 153
- UI.properties ...................................................................... 155
- AdditionalLabels.properties .............................................. 158
- Additional property files ................................................... 158

© Copyright IBM Corp. 2004
Chapter 10. Configuring System Properties

This chapter provides detailed information about the property keys and values contained in the Tivoli Identity Manager enRole.properties system configuration file.

Section topics:

- “Understanding Properties Files” on page 92

enRole.properties file reference by section:

- “WebLogic-specific Configuration” on page 93
- “WebSphere-specific Configuration” on page 97
- “Application Server Information” on page 101
- “Default Tenant Information” on page 103
- “LDAP Server Information” on page 104
- “Cache Information” on page 105
- “Messaging Information” on page 106
- “Scheduling Information” on page 109
- “Password Transaction Monitor Settings” on page 110
- “XML and DTD Information” on page 111
- “LDAP Connection Pool Information” on page 112
- “Encryption Information” on page 113
- “System Configuration Program” on page 114
- “Workflow Configuration Information” on page 115
- “Mail Services Configuration” on page 119
- “Reconciliation Information” on page 120
- “Shared Secret Hashing” on page 122
- “SSL Two-way Authentication Properties” on page 123
- “Request Management UI Configuration” on page 124
- “Application Client Request Configuration” on page 125
Understanding Properties Files

Java properties files define attributes that allow customizing and control of the Java software. Standard system properties files and custom properties files are used to configure user preferences and user customizations. A Java properties file defines the values of named resources that can specify program options such as database access information, environment settings, and special features and functionality.

A properties file defines named resources using a property key and value pair format:

```
property-key-name = value
```

The *property-key-name* is an identifier for the resource. The *value* is the name of the actual Java object that provides the resource.

Tivoli Identity Manager uses a number of properties files to control the functionality of the program and to allow user customization of special features.
WebLogic-specific Configuration

The following properties configure values specific to integration of Tivoli Identity Manager with the WebLogic application server:

<table>
<thead>
<tr>
<th>Platform Context Factory Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.platform.contextFactory</code></td>
</tr>
<tr>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td>Specifies the Java class for the platform context factory that defines the integration point for Tivoli Identity Manager with the WebLogic application server.</td>
</tr>
<tr>
<td>Default (entered as one line):</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application server</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.appServer.contextFactory</code></td>
</tr>
<tr>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td>Specifies the Java class that determines what JNDI factory to use with the WebLogic application server.</td>
</tr>
<tr>
<td>Default:</td>
</tr>
<tr>
<td><code>enrole.appServer.contextFactory = weblogic.jndi.WLInitialContextFactory</code></td>
</tr>
</tbody>
</table>

| `enrole.appServer.url.redirect` |
| Do not modify this property key and value. |
| Specifies the URL required to direct requests to the Tivoli Identity Manager Server. |
| Example (default): |
| `enrole.appServer.url.redirect = /enrole` |

| `enrole.appServer.url` |
| This property key and value should be changed only by a qualified administrator. |
| Specifies the location of the application server naming service. This value is obtained during Tivoli Identity Manager installation. |
| Example: |
| `enrole.appServer.url = t3://localhost` |

| `enrole.appServer.pwdKey` |
| NOT USED. |

| `enrole.appServer.systemUser` |
### enrole.appServer.systemUser

This property key and value should be changed only by a qualified administrator. Modify using the `runConfig` utility only.

Specifies the name of the administrator user for WebLogic. In a WebLogic environment, this value is required and the property must always be considered enabled. The value is used to start, stop, and configure the Tivoli Identity Manager server. The value is also used by Tivoli Identity Manager installation and configuration routines to authenticate to WebLogic.

Example:
```
enrole.appServer.systemUser = system
```

### enrole.appServer.systemUser.credentials

This property key and value should be changed only by a qualified administrator. Modify using the `runConfig` utility only.

Specifies the password for the `systemUser`.

Example:
```
enrole.appServer.systemUser.credentials = enrole
```

### enrole.appServer.ejbuser.principal

This property key and value should be changed only by a qualified administrator. Modify using the `runConfig` utility only.

 Specifies the name used by Tivoli Identity Manager to authenticate when making calls on Java beans.

Example:
```
enrole.appServer.ejbuser.principal = rasweb
```

### enrole.appServer.ejbuser.credentials

This property key and value should be changed only by a qualified administrator. Modify using the `runConfig` utility only.

Specifies the password for the `ejbuser`.

Encryption of this value is specified by the `enrole.password.appServer.encrypted` property in `enrole.properties`.

Example:
```
enrole.appServer.ejbuser.credentials = password
```

### enrole.appServer.usertransaction.jndiname

Do not modify this property key and value.

Specifies the JNDI name of the JTA (Java Transaction API) User Transaction object.

Default:
```
enrole.appServer.usertransaction.jndiname = javax.transaction.UserTransaction
```

### enrole.appServer.name.java.option

Do not modify this property key and value.

Specifies JVM options when starting the WebLogic server.

Example (default):
```
enrole.appServer.name.java.option = weblogic.Name
| **Application server servlet path separator** |
| **enrole.servlet.path.separator** |
| Do not modify this property key and value. |
| Specifies the separator character used to specify path names to required resources. |
| Default (WebLogic): |
| `enrole.servlet.path.separator = /` |

**Messaging**

| **enrole.messaging.JMSServerUrl** |
| This property key and value should be changed only by a qualified administrator. |
| Specifies the location of the naming service that contains the Java Messaging Service (JMS). |
| For WebSphere, this value is the same as `enrole.appServer.url`. |
| Example: |
| `enrole.messaging.JMSServerUrl = t3://localhost` |

| **enrole.messaging.sessionPoolFactory** |
| NOT USED. |

| **enrole.messaging.weblogic.sessionPoolFactory** |
| NOT USED. |

**Login helper**

| **enrole.appServer.loginHelper.class** |
| NOT USED. |

**Event notification system login**

| **SystemLoginContextFactory** |
| Do not modify this property key and value. |
| Specifies the Java factory class for event notification system login appropriate for WebLogic. |
| Default (entered as one line): |
| `SystemLoginContextFactory = com.ibm.itim.remoteservices.provider.itim. weblogic.WLSystemLoginContextFactory` |

**Messaging queue attributes**
• **MAX_THREADS**
  Specifies the maximum number of listener threads for this queue. If this attribute is not set, the value used is the default set by `enrole.messaging.defaultMaxThreads`. If this attribute is set, the value specified should not be greater than the `enrole.messaging.defaultMaxThreads` value.

• **MIN_THREADS**
  Specifies the minimum number of listener threads for this queue. If this attribute is not set, the default value used is 10. If this attribute is set, the specified value should not be less than the `enrole.messaging.minThreads` value.

• **OVERCAPACITY_WAIT_TIME**
  Time to wait to receive a new message when the system is over capacity (messages are waiting to be processed). Default is 60 seconds. This attribute is only appropriate for the `workflowPendingQueue`.

• **PRIORITY**
  The thread priority. Refer to JVM documentation for information on selecting appropriate values (1 – 5). Default is 1. Setting all queues to the same value is recommended.

• **RECEIVE_TIMEOUT**
  Time (in seconds) to wait for a response from the JMS server that it can receive a message. Default is 60 seconds.

• **WAIT_TIME**
  Time (in seconds) to wait to process a new message from the queue. If the value is 0, then there is no wait to process a new message. Default is 0 seconds.

• **TRANSACTED**
  True – support user transactions
  False – do not support user transactions

Examples:
```
enrole.messaging.adhocSyncQueue.attributes = TRANSACTED=true RECEIVE_TIMEOUT=1 MAX_THREADS=5 MIN_THREADS=1
enrole.messaging.workflowQueue.attributes = TRANSACTED=true RECEIVE_TIMEOUT=1 MAX_THREADS=10 MIN_THREADS=5
enrole.messaging.workflowPendingQueue.attributes = TRANSACTED=true RECEIVE_TIMEOUT=1 WAIT_TIME=0 OVERCAPACITY_WAIT_TIME=10 MAX_THREADS=5 MIN_THREADS=2
enrole.messaging.workflowAbortQueue.attributes = TRANSACTED=true RECEIVE_TIMEOUT=1 WAIT_TIME=0 MAX_THREADS=5 MIN_THREADS=1
enrole.messaging.remoteServicesQueue.attributes = TRANSACTED=false RECEIVE_TIMEOUT=1 WAIT_TIME=0 MAX_THREADS=7 MIN_THREADS=2
enrole.messaging.mailServicesQueue.attributes = TRANSACTED=false RECEIVE_TIMEOUT=1 WAIT_TIME=0 MAX_THREADS=3 MIN_THREADS=1
```
WebSphere-specific Configuration

The following properties configure values specific to integration of Tivoli Identity Manager with the WebSphere application server:

<table>
<thead>
<tr>
<th>Platform Context Factory Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.platform.contextFactory</td>
</tr>
<tr>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td>Specifies the Java class for the platform context factory that defines the integration point for Tivoli Identity Manager with the WebSphere application server.</td>
</tr>
<tr>
<td>Default (entered as one line): enrole.platform.contextFactory = com.ibm.itim.apps.impl.websphere.WebSpherePlatformContextFactory</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application server</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.appServer.contextFactory</td>
</tr>
<tr>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td>Specifies the Java class that determines what JNDI factory to use with the WebSphere application server.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enrole.appServer.url</th>
</tr>
</thead>
<tbody>
<tr>
<td>This property key and value should be changed only by a qualified administrator.</td>
</tr>
<tr>
<td>Specifies the location of the application server naming service. This value is obtained during Tivoli Identity Manager installation.</td>
</tr>
<tr>
<td>Example: enrole.appServer.url = iiop://localhost:2809</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enrole.appServer.usertransaction.jndiname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td>Specifies the JNDI name of the JTA (Java Transaction API) User Transaction object.</td>
</tr>
<tr>
<td>Default: enrole.appServer.usertransaction.jndiname = jta/usertransaction</td>
</tr>
</tbody>
</table>

<p>| enrole.appServer.systemUser                   |</p>
<table>
<thead>
<tr>
<th>Property Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.appServer.systemUser</code></td>
<td>Specifies the name of the administrator user for WebSphere when security is enabled. The value is required only when global security is enabled. The value is not set if security is not enabled. The value is used to start, stop, and configure the Tivoli Identity Manager server. The value is also used by Tivoli Identity Manager installation and configuration routines to authenticate to WebSphere.</td>
</tr>
<tr>
<td><code>enrole.appServer.systemUser.credentials</code></td>
<td>Example: <code>enrole.appServer.systemUser = system</code></td>
</tr>
<tr>
<td><code>enrole.appServer.ejbuser.principal</code></td>
<td>Specifies the name used by Tivoli Identity Manager to authenticate when making calls on Java beans.</td>
</tr>
<tr>
<td><code>enrole.appServer.ejbuser.principal</code></td>
<td>Example: <code>enrole.appServer.ejbuser.principal = rasweb</code></td>
</tr>
<tr>
<td><code>enrole.appServer.ejbuser.credentials</code></td>
<td>Specifies the password for the <code>ejbuser</code>.</td>
</tr>
<tr>
<td><code>enrole.appServer.ejbuser.credentials</code></td>
<td>Example: <code>enrole.appServer.ejbuser.credentials = password</code></td>
</tr>
<tr>
<td><code>enrole.servlet.path.separator</code></td>
<td>Specifies the separator character used to specify path names to required resources. The value is used as a separator in string concatenation for static JMS server URL creation.</td>
</tr>
<tr>
<td><code>enrole.servlet.path.separator</code></td>
<td>Default (WebSphere): <code>enrole.servlet.path.separator = .</code></td>
</tr>
</tbody>
</table>

**Messaging**

<table>
<thead>
<tr>
<th>Property Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.messaging.JMSServerUrl</code></td>
<td></td>
</tr>
</tbody>
</table>
This property key and value should be changed only by a qualified administrator.
Specifies the location of the naming service that contains the Java Messaging Service (JMS).

For WebSphere, this value is the same as enrole.appServer.url.

Example:
enrole.messaging.JMSServerUrl = iiop://localhost:2809

**Login helper**
enrole.appServer.loginHelper.class

Do not modify this property key and value.
Specifies the Java class used to log each thread into J2EE Security.
Default:
enrole.appServer.loginHelper.class = com.ibm.itim.util.was.WAS40LoginHelper

**Workflow tier URL**
enrole.wfcluster.url

Do not modify this property key and value.
This property is valid for WebSphere only.
Specifies where the workflow tier beans can be looked up. Specify a value for this only on application tier machines of a Functional Cluster. Leave the default value as is for all other cases.
Example (default):
enrole.wfcluster.url = iiop://localhost:2809/cell/clusters/WFCluster

**Event notification system login**
SystemLoginContextFactory

Do not modify this property key and value.
Specifies the Java factory class for event notification system login appropriate for WebSphere.
Default (entered as one line):
SystemLoginContextFactory = com.ibm.itim.remoteservices.provider.itim.websphere.WSSystemLogonContextFactory

**Messaging queue attributes**
- **MAX_THREADS**
  Specifies the maximum number of listener threads for this queue. If this attribute is not set, the value used is the default set by `enrole.messaging.defaultMaxThreads`. If this attribute is set, the value specified should not be greater than the `enrole.messaging.defaultMaxThreads` value.

- **MIN_THREADS**
  Specifies the minimum number of listener threads for this queue. If this attribute is not set, the default value used is 10. If this attribute is set, the specified value should not be less than the `enrole.messaging.minThreads` value.

- **OVERCAPACITY_WAIT_TIME**
  Time to wait to receive a new message when the system is over capacity (messages are waiting to be processed). Default is 60 seconds. This attribute is only appropriate for the `workflowPendingQueue`.

- **PRIORITY**
  The thread priority. Refer to JVM documentation for information on selecting appropriate values (1 – 5). Default is 1. Setting all queues to the same value is recommended.

- **RECEIVE_TIMEOUT**
  Time (in seconds) to wait for a response from the JMS server that it can receive a message. Default is 60 seconds.

- **WAIT_TIME**
  Time (in seconds) to wait to process a new message from the queue. If the value is 0, then there is no wait to process a new message. Default is 0 seconds.

- **TRANSACTED**
  True – support user transactions
  False – do not support user transactions

Examples:

```
enrole.messaging.adhocSyncQueue.attributes = TRANSACTED=true
  RECEIVE_TIMEOUT=60 MAX_THREADS=5 MIN_THREADS=1

enrole.messaging.workflowQueue.attributes = TRANSACTED=true
  RECEIVE_TIMEOUT=60 MAX_THREADS=10 MIN_THREADS=5

enrole.messaging.workflowPendingQueue.attributes = TRANSACTED=true
  RECEIVE_TIMEOUT=60 WAIT_TIME=0 OVERCAPACITY_WAIT_TIME=10 MAX_THREADS=5
  MIN_THREADS=2

enrole.messaging.workflowAbortQueue.attributes = TRANSACTED=true
  RECEIVE_TIMEOUT=60 WAIT_TIME=0 MAX_THREADS=5 MIN_THREADS=1

enrole.messaging.remoteServicesQueue.attributes = TRANSACTED=false
  RECEIVE_TIMEOUT=60 WAIT_TIME=0 MAX_THREADS=7 MIN_THREADS=2

enrole.messaging.mailServicesQueue.attributes = TRANSACTED=false
  RECEIVE_TIMEOUT=60 WAIT_TIME=0 MAX_THREADS=3 MIN_THREADS=1
```
## Application Server Information

The following properties configure values specific to the application server (such as WebSphere or WebLogic) using Tivoli Identity Manager.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User-selected locale</strong></td>
<td>Specifies the locale setting for the Tivoli Identity Manager environment.</td>
<td>locale = en</td>
</tr>
<tr>
<td><strong>Context factory name</strong></td>
<td>Specifies the unique name of the application server.</td>
<td>enrole.appServer.name = myserver</td>
</tr>
<tr>
<td></td>
<td>In a cluster environment, it is important that this name be unique for each member in the cluster.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example (default):</td>
<td></td>
</tr>
<tr>
<td></td>
<td>enrole.appServer.name</td>
<td></td>
</tr>
<tr>
<td><strong>enrole.appServer.config.latency</strong></td>
<td>NOT USED</td>
<td></td>
</tr>
<tr>
<td><strong>enrole.password.database.encrypted</strong></td>
<td>Use the runConfig utility to modify this property.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifies whether or not the password for the database connection (specified by database.db.password in enroleDatabase.properties) is encrypted. Valid values are:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• true – encrypted</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• false – not encrypted</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example (default):</td>
<td></td>
</tr>
<tr>
<td></td>
<td>enrole.password.database.encrypted = false</td>
<td></td>
</tr>
<tr>
<td><strong>enrole.password.ldap.encrypted</strong></td>
<td>Use the runConfig utility to modify this property.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifies whether or not the LDAP password (specified by the java.naming.security.credentials property in enRoleLDAPConnection.properties) is encrypted. Valid values are:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• true – encrypted</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• false – not encrypted</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example (default):</td>
<td></td>
</tr>
<tr>
<td></td>
<td>enrole.password.ldap.encrypted = false</td>
<td></td>
</tr>
<tr>
<td><strong>enrole.password.appServer.encrypted</strong></td>
<td>Use the runConfig utility to modify this property.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifies whether or not the password for the application server (specified by the database.db.password in enroleDatabase.properties) is encrypted. Valid values are:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• true – encrypted</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• false – not encrypted</td>
<td></td>
</tr>
</tbody>
</table>
Use the `runConfig` utility to modify this property.

Specifies whether or not the application server password (specified by the `enrole.appServer.ejbuser.credentials` property in `enrole.properties`) is encrypted. Valid values are:

- true – encrypted
- false – not encrypted

Example (default):

```
enrole.password.appServer.encrypted = false
```
## Default Tenant Information

The following properties configure the details of the organization name used by the directory server.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.defaulttenant.id</td>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td></td>
<td>Specifies the short format of the organization name used by the directory server.</td>
</tr>
<tr>
<td></td>
<td>This value is specified during installation of Tivoli Identity Manager.</td>
</tr>
<tr>
<td></td>
<td>Example (default):</td>
</tr>
<tr>
<td></td>
<td>enrole.defaulttenant.id = Tivoli</td>
</tr>
<tr>
<td></td>
<td>In LDAP, this value would be expressed as:</td>
</tr>
<tr>
<td></td>
<td>ou = Tivoli</td>
</tr>
<tr>
<td>enrole.organization.name</td>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td></td>
<td>Specifies the long format of the organization name used by the directory server.</td>
</tr>
<tr>
<td></td>
<td>This value is specified during installation of Tivoli Identity Manager.</td>
</tr>
<tr>
<td></td>
<td>Example (default):</td>
</tr>
<tr>
<td></td>
<td>enrole.organization.name = Tivoli</td>
</tr>
</tbody>
</table>
# LDAP Server Information

The following properties configure values used by the Tivoli Identity Manager directory server.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.ldapserver.root</td>
<td>Specifies the top level domain entry node of the directory server data structure (dc = domain control). This value is specified during installation of Tivoli Identity Manager. Use the runConfig utility to modify this value. Example (default): enrole.ldapserver.root = dc=com</td>
</tr>
<tr>
<td>enrole.ldapserver.home</td>
<td>Do not modify this property key and value. Specifies the location of system configuration information in the Tivoli Identity Manager directory server. Default: enrole.ldapserver.home = ou=itim</td>
</tr>
<tr>
<td>enrole.ldapserver.agelimit</td>
<td>Specifies the number of days an object remains in the system’s recycle bin before it becomes available for deletion by manually-run cleanup scripts. The recycle bin age limit protects objects in the recycle bin from cleanup scripts for the specified length of time. Cleanup scripts can only remove those objects that are older than the age limit setting. For example, if the age limit setting is 62 days (default), only objects older than 62 days (have been in the recycle bin for more than 62 days) can be deleted by manually-run cleanup scripts. This property key and value should be changed only by a qualified administrator. Use the runConfig utility to modify this value. Example (default): enrole.ldapserver.agelimit = 62</td>
</tr>
<tr>
<td>enrole.ldapserver.ditlayout</td>
<td>Specifies the Java class that defines the structure of the data stored in the directory server. Default – flat structure: enrole.ldapserver.ditlayout = com.ibm.itim.dataservices.dit.itim.FlatHashedLayout</td>
</tr>
<tr>
<td>enrole.ldap.provider</td>
<td>NOT USED</td>
</tr>
</tbody>
</table>
## Cache Information

The following properties configure values affecting the performance of the system cache.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.profile.timeout</code></td>
<td>This property key and value affects performance tuning for Tivoli Identity Manager and should be changed only by a qualified administrator.</td>
</tr>
<tr>
<td></td>
<td>Specifies the timeout value for information residing in the profile section of the cache. Information exceeding this timeout value is removed from the cache.</td>
</tr>
<tr>
<td></td>
<td>Value is expressed in minutes.</td>
</tr>
<tr>
<td>Example (default):</td>
<td><code>enrole.profile.timeout = 10</code></td>
</tr>
<tr>
<td><code>enrole.schema.timeout</code></td>
<td>This property key and value affects performance tuning for Tivoli Identity Manager and should be changed only by a qualified administrator.</td>
</tr>
<tr>
<td></td>
<td>Specifies the timeout value for information residing in the schema section of the cache. Information exceeding this timeout value is removed from the cache.</td>
</tr>
<tr>
<td></td>
<td>Value is expressed in minutes.</td>
</tr>
<tr>
<td>Example (default):</td>
<td><code>enrole.schema.timeout = 10</code></td>
</tr>
</tbody>
</table>
**Messaging Information**

The following properties configure the internal communication between components of the Java Message Service (JMS) used by Tivoli Identity Manager. The adjustment of these property values are important to proper performance tuning and scalability of the Tivoli Identity Manager product. Property values in this section should only be changed by qualified administrators.

### Connection factory configuration

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.messaging.queueConnectionFactory</td>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td></td>
<td>Specifies the name of the connection factory for Java Naming and Directory Interface (JNDI) queue connections.</td>
</tr>
<tr>
<td></td>
<td>This property value is built-in and must not be changed or removed. The value is crucial to the successful configuration of the corresponding application server queue connection.</td>
</tr>
<tr>
<td>Example (default):</td>
<td>enrole.messaging.queueConnectionFactory = enrole.jms.QueueConnectionFactory</td>
</tr>
</tbody>
</table>

### Global listener threads configuration

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.messaging.defaultMaxThreads</td>
<td>This property key and value affects performance tuning for Tivoli Identity Manager and should be changed only by a qualified administrator.</td>
</tr>
<tr>
<td></td>
<td>Specifies the default value for the maximum number of listener threads kept in the connection pool for each JMS queue.</td>
</tr>
<tr>
<td></td>
<td>This default value is used if the MAX_THREADS attribute is not set on a queue (see Messaging queue configuration section).</td>
</tr>
<tr>
<td>Example (default):</td>
<td>enrole.messaging.defaultMaxThreads = 10</td>
</tr>
<tr>
<td>enrole.messaging.minThreads</td>
<td>This property key and value affects performance tuning for Tivoli Identity Manager and should be changed only by a qualified administrator.</td>
</tr>
<tr>
<td></td>
<td>Specifies the minimum number of listener threads allowed for each JMS queue. The actual number of JMS listener threads cannot be less than this threshold value.</td>
</tr>
<tr>
<td></td>
<td>This global value can be overridden for individual queues by the MIN_THREADS attribute (which is currently hard-coded at 10).</td>
</tr>
<tr>
<td>Example (default):</td>
<td>enrole.messaging.minThreads = 1</td>
</tr>
<tr>
<td>enrole.messaging.maxThreads</td>
<td></td>
</tr>
</tbody>
</table>

---

IBM Tivoli Identity Manager: Configuration Guide
This property key and value affects performance tuning for Tivoli Identity Manager and should be changed only by a qualified administrator.

Specifies the maximum number of listener threads allowed for each JMS queue. The actual number of JMS listener threads cannot be greater than this threshold value.

This global value sets the threshold limit for the allowable value of the MAX_THREADS queue attribute.

Example (default):
enrole.messaging.maxThreads = 500

---

**Message timeout configuration**

enrole.messaging.ttl

This property key and value affects performance tuning for the JMS and should be changed only by a qualified administrator.

Specifies the lifetime of a message in the queue in minutes.

Example (default):
enrole.messaging.ttl = 1440

---

enrole.messaging.timeout

This property key and value affects performance tuning for the JMS and should be changed only by a qualified administrator.

Specifies the user transaction timeout for message processing (in seconds).

Suggested default is 360 seconds.

This value is crucial to system performance and should adjusted only after careful observation of the behavior of an individual deployment. A low setting on a busy system will cause message processing timeouts. A high setting can allow a single lengthy process to deadlock the entire system.

Example (suggested initial setting):
enrole.messaging.timeout = 360

---

**Message processing control**

enrole.messaging.threshold

This property key and value affects performance tuning for Tivoli Identity Manager and should be changed only by a qualified administrator.

Specifies the percent level of simultaneous processing of messages (message load).

A value less than 100% creates a "first message in, first message out" algorithm. A value greater than 100% allows all messages in at once for processing, but can have the consequence of placing stress on the system.

Example (default percent):
enrole.messaging.threshold = 60

---

**Message system initialization**

enrole.messaging.QueueLookupRetryCount
This property key and value affects performance tuning for Tivoli Identity Manager and should be changed only by a qualified administrator.

Specifies the number of retries by Tivoli Identity Manager to establish connections to application server queues.

Example (default):
```
enrole.messaging.QueueLookupRetryCount = 5
```

### enrole.messaging.QueueLookupInterval

This property key and value affects performance tuning for Tivoli Identity Manager and should be changed only by a qualified administrator.

Specifies the interval (in seconds) between queue connection retries.

Example (default):
```
enrole.messaging.QueueLookupInterval = 60
```

### Messaging queue configuration

```
enrole.messaging.managers= \
  enrole.messaging.adhocSyncQueue \n  enrole.messaging.workflowQueue \n  enrole.messaging.workflowPendingQueue \n  enrole.messaging.remoteServicesQueue \n  enrole.messaging.mailServicesQueue \n  enrole.messaging.workflowAbortQueue
```

Do not modify these property keys and values.

Specifies the key names of supported Tivoli Identity Manager queues.

```
enrole.messaging.adhocSyncQueue=adhocSyncQueue
enrole.messaging.workflowQueue=workflowQueue
enrole.messaging.workflowPendingQueue=workflowPendingQueue
enrole.messaging.remoteServicesQueue=remoteServicesQueue
enrole.messaging.mailServicesQueue=mailServicesQueue
enrole.messaging.workflowAbortQueue=workflowAbortQueue
```

Do not modify these property keys and values.

Specifies the actual queue name as referenced by the application server.
Scheduling Information

The following properties configure the internal scheduler responsible for running calendar-based scheduled events. Events and their schedules are stored in a database table.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example (default):</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.scheduling.heartbeat</td>
<td>This property key and value affects performance tuning for Tivoli Identity Manager and should be changed only by a qualified administrator. Specifies how often the event monitor checks the database table for scheduled events. Value is expressed in seconds.</td>
<td>enrole.scheduling.heartbeat = 60</td>
</tr>
<tr>
<td>enrole.scheduling.timeout</td>
<td>Specifies the timeout value for the event processor. Value is expressed in minutes.</td>
<td>enrole.scheduling.timeout = 10</td>
</tr>
<tr>
<td>enrole.scheduling.fetchsize</td>
<td>Specifies the number of messages to retrieve at a time when in batch mode.</td>
<td>enrole.scheduling.fetchsize = 50</td>
</tr>
</tbody>
</table>
Password Transaction Monitor Settings

The following property is a configuration setting for the password transaction monitor.

When a password for a user is changed or automatically generated, an e-mail notification is sent to the user. The e-mail contains either the actual password or a link that the user can follow to obtain the new password. This activity is called a password transaction. The user must respond to the e-mail and incorporate the new password within a specified amount of time. If the user fails to respond within the allowed time period, the password transaction expires.

The password transaction monitor is responsible for checking responses to password transactions and expiring those transactions where the user has failed to respond to the e-mail notification.

<table>
<thead>
<tr>
<th>enrole.passwordtransactionmonitor.heartbeat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies how often the password transaction monitor checks for expired password transactions.</td>
</tr>
<tr>
<td>Value is expressed in hours.</td>
</tr>
<tr>
<td>Example (default):</td>
</tr>
<tr>
<td>enrole.passwordtransactionmonitor.heartbeat = 1</td>
</tr>
</tbody>
</table>
XML and DTD Information

This section is no longer used.

<table>
<thead>
<tr>
<th>enrole.dtd.uri</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOT USED</td>
</tr>
</tbody>
</table>

Chapter 10. Configuring System Properties  111
**LDAP Connection Pool Information**

The following properties configure values affecting cache connection requests to the Tivoli Identity Manager directory server.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.connectionpool.maxpoolsize</code></td>
<td>This property key and value affects performance tuning for Tivoli Identity Manager and should be changed only by a qualified administrator. Specifies the maximum number of physical LDAP connections that can be created.</td>
<td><code>enrole.connectionpool.maxpoolsize = 100</code></td>
</tr>
<tr>
<td><code>enrole.connectionpool.initialpoolsize</code></td>
<td>This property key and value affects performance tuning for Tivoli Identity Manager and should be changed only by a qualified administrator. Specifies the initial number of physical LDAP connections to create for the LDAP connection pool. This value must be less than or equal to the <code>maxpoolsize</code> value.</td>
<td><code>enrole.connectionpool.initialpoolsize = 50</code></td>
</tr>
<tr>
<td><code>enrole.connectionpool.incrementcount</code></td>
<td>This property key and value affects performance tuning for Tivoli Identity Manager and should be changed only by a qualified administrator. Specifies the number of connections created any time the LDAP connection pool is expanded (incremented) to accommodate increasing demand.</td>
<td><code>enrole.connectionpool.incrementcount = 3</code></td>
</tr>
</tbody>
</table>
Encryption Information

The following properties configure values affecting encryption of passwords.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.encryption.algorithm</td>
<td>Do not modify this property key and value. Specifies the cipher suite to use for encryption.</td>
</tr>
<tr>
<td>Default:</td>
<td>enrole.encryption.algorithm = PBE/SHA1/RC2/CBC/PKCS12PBE-5-128</td>
</tr>
<tr>
<td>enrole.encryption.password</td>
<td>Do not modify this property key and value. Specifies the password used as an input parameter for Password-Based Encryption (PBE), a method of encrypting and decrypting data using a secret key based on a user-supplied password. This value is specified during Tivoli Identity Manager installation. Default: enrole.encryption.password = sunshine</td>
</tr>
<tr>
<td>enrole.encryption.passwordDigest</td>
<td>Do not modify this property key and value. Specifies the type of password digest used for a Tivoli Identity Manager password. Choices are: “SHA” and “MD5” • SHA – The Secure Hash Algorithm, as defined in Secure Hash Standard, NIST FIPS 180-1 • MD5 – The MD5 message digest algorithm as defined in RFC 1321 Default: enrole.encryption.passwordDigest = MD5</td>
</tr>
</tbody>
</table>
## System Configuration Program

The following properties configure listening port settings for the Tivoli Identity Manager system.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.system.listenPort</code></td>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td></td>
<td>Specifies the TCP (non-secure communication) listening port value.</td>
</tr>
<tr>
<td></td>
<td>This value is set during Tivoli Identity Manager installation.</td>
</tr>
<tr>
<td></td>
<td>Example (default): <code>enrole.system.listenPort = 80</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.system.SSLlistenPort</code></td>
<td>Do not modify this property key and value.</td>
</tr>
<tr>
<td></td>
<td>Specifies the SSL (secure communication) listening port value.</td>
</tr>
<tr>
<td></td>
<td>This value is set during Tivoli Identity Manager installation.</td>
</tr>
<tr>
<td></td>
<td>Example (default): <code>enrole.system.SSLlistenPort = 443</code></td>
</tr>
</tbody>
</table>
Workflow Configuration Information

The following properties configure the core Tivoli Identity Manager workflow engine.

### Workflow configuration

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.workflow.processcache</td>
<td>Enables/disables internal process caching.</td>
</tr>
<tr>
<td></td>
<td>In single-server environments, the value can be &quot;true&quot;; in cluster environments, the value is always &quot;false&quot;.</td>
</tr>
<tr>
<td></td>
<td>Caching is not supported in cluster deployments. The value for this property is automatically set to &quot;false&quot; (disabled) during installation of a cluster configuration.</td>
</tr>
<tr>
<td></td>
<td>Example (default single-server): enrole.workflow.processcache = true</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.workflow.notifyoption</td>
<td>Specifies options for the notification of process completions.</td>
</tr>
<tr>
<td></td>
<td>A value of 1 means the requestor will be notified when the workflow completes. A value of 0 means the requestor will not be notified.</td>
</tr>
<tr>
<td></td>
<td>Example (default): enrole.workflow.notifyoption = 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.workflow.notifypassword</td>
<td>Specifies the type of e-mail notification in a password transaction (caused when a user password is changed or automatically generated).</td>
</tr>
<tr>
<td></td>
<td>• true – e-mail notification of a password change can be sent to the user. The actual notification mechanism, including the inclusion, or not, of the actual password in the e-mail, is dictated by the configuration of the enrole.workflow.notification.newpassword property value in the enrole.properties configuration file.</td>
</tr>
<tr>
<td></td>
<td>• false – e-mail notification of a password change is sent to the user. The e-mail contains a URL where the user can obtain the password. The URL prompts the user for the user’s shared secret.</td>
</tr>
<tr>
<td></td>
<td>Example (default): enrole.workflow.notifypassword = true</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.workflow.maxasyncactivitycreate</td>
<td>NOT USED</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.workflow.maxretry</td>
<td>Specifies the number of times an attempt is made to re-invoke a workflow application that has initially failed. See enrole.workflow.retrydelay.</td>
</tr>
<tr>
<td></td>
<td>Example (default): enrole.workflow.maxretry = 2</td>
</tr>
</tbody>
</table>
### enrole.workflow.retrydelay

This property key and value should be changed only by a qualified administrator.

Specifies the time delay period between successive retries to re-invoke a workflow application that has initially failed. See enrole.workflow.maxretry.

Value is expressed in milliseconds.

Example (default):

```
enrole.workflow.retrydelay = 60000
```

### enrole.workflow.skipapprovalforrequester

This property key and value should be changed only by a qualified administrator.

For a workflow activity that requires approval, specifies whether to skip the approval for other approvers if the requestor is also an approver. The requestor is also skipped from approval by design.

A value of "true" skips approval for other approvers (if the requestor is one of the approvers).

A value of "false" forces approval checking from all other required approvers of the activity except the requestor (if the requestor is also an approver).

Example (default):

```
enrole.workflow.skipapprovalforrequester = false
```

### enrole.workflow.skipfornoncompliantaccount

This property key and value should be changed only by a qualified administrator.

Specifies whether or not to engage the entitlement workflow associated with the account when a system account modification is triggered as a result of a policy enforcement action.

A value of "true" means skip this action.

A value of "false" means do not skip this action.

Example (default):

```
enrole.workflow.skipfornoncompliantaccount = true
```

### Workflow notification configuration

This section allows you to introduce custom Java class files that customize the implementation of workflow notification types.

Refer to the following documentation for information on how to customize notifications:

```
<install-dir>\extensions\doc\mail\mail.html
```

### enrole.workflow.notification.activitytimeout

This property key and value should be changed only by a qualified administrator.

Specifies the default Java class that generates the workflow activity timeout notification.

Example (built-in default) (entered as one line):

```
enrole.workflow.notification.activitytimeout = com.ibm.itim.workflow.
notification.ActivityTimeoutNotification
```

### enrole.workflow.notification.processtimeout
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example (built-in default)</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.workflow.notification.processTimeout</td>
<td>This property key and value should be changed only by a qualified administrator. Specifies the default Java class that generates the workflow process timeout notification.</td>
<td>enrole.workflow.notification.processTimeout = com.ibm.itim.workflow. notification.ProcessTimeoutNotification</td>
</tr>
<tr>
<td>enrole.workflow.notification.processComplete</td>
<td>This property key and value should be changed only by a qualified administrator. Specifies the default Java class that generates the workflow process complete notification.</td>
<td>enrole.workflow.notification.processComplete = com.ibm.itim.workflow. notification.ProcessCompleteNotification</td>
</tr>
<tr>
<td>enrole.workflow.notification.pendingWork</td>
<td>This property key and value should be changed only by a qualified administrator. Specifies the default Java class that generates the workflow pending work notification.</td>
<td>enrole.workflow.notification.pendingWork = com.ibm.itim.workflow. notification.PendingWorkNotification</td>
</tr>
<tr>
<td>enrole.workflow.notification.newAccount</td>
<td>This property key and value should be changed only by a qualified administrator. Specifies the default Java class that generates the workflow new account notification.</td>
<td>enrole.workflow.notification.newAccount = com.ibm.itim.workflow. notification.NewAccountNotification</td>
</tr>
<tr>
<td>enrole.workflow.notification.newPassword</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This property key and value should be changed only by a qualified administrator.

Specifies the default Java class that generates an e-mail notification to a user when that user’s password is changed. This property is activated when:

```
enrole.workflow.notifyPassword = true
```

This workflow responds for the following three password change scenarios:

- When the user makes the password change on the account
- When the administrator forces a password change on the account
- When the user is successfully identified through the password challenge/response feature, and challenge/response is configured for Email Password.

Available classes include:

- **NewPasswordNotification**
  E-mail notification that includes the password (in ASCII text) is sent to the user (default).

- **EmptyNotificationFactory**
  Suppresses e-mail notification.

- **PasswordChangeNotificationFactory**
  E-mail notification that does not include the password is sent to the user. Message body says: “Process completed”.

Example (built-in default) (entered as one line):

```
enrole.workflow.notification.newpassword = com.ibm.itim.workflow.notification.NewPasswordNotification
```

```
enrole.workflow.notification.deprovision
```

This property key and value should be changed only by a qualified administrator.

Specifies the default Java class that generates the workflow deprovisioning notification.

Example (built-in default) (entered as one line):

```
enrole.workflow.notification.deprovision = com.ibm.itim.workflow.notification.DeprovisionNotification
```

```
enrole.workflow.notification.workorder
```

This property key and value should be changed only by a qualified administrator.

Specifies the default Java class that generates the workflow work order notification.

Example (built-in default) (entered as one line):

```
enrole.workflow.notification.workorder = com.ibm.itim.workflow.notification.WorkOrderNotification
```
Mail Services Configuration

The following property is a configuration setting for internal mail notification.

<table>
<thead>
<tr>
<th>enrole.mail.notify</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies whether the sending of workflow internal e-mail is synchronized or asynchronized.</td>
</tr>
<tr>
<td>Values include: &quot;SYNC&quot; and &quot;ASYNC&quot;. &quot;SYNC&quot; means synchronized and &quot;ASYNC&quot; means asynchronized.</td>
</tr>
<tr>
<td>Example (default):</td>
</tr>
<tr>
<td>enrole.mail.notify = ASYNC</td>
</tr>
</tbody>
</table>
## Reconciliation Information

The following properties configure values affecting the reconciliation process where data retrieved from agents is synchronized in the Tivoli Identity Manager database.

### Reconciliation configuration

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example (default):</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.reconciliation.accountcachesize</td>
<td>This property key and value should be changed only by a qualified administrator. Specifies the maximum size of the existing accounts cache (number of accounts) used for the reconciliation process.</td>
<td>enrole.reconciliation.accountcachesize = 2000</td>
</tr>
<tr>
<td>enrole.reconciliation.threadcount</td>
<td></td>
<td>enrole.reconciliation.threadcount = 8</td>
</tr>
</tbody>
</table>

### Unsolicited notification events

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>account.EventProcessorFactory</td>
<td>Do not modify this property key and value. Specifies the built-in Java class for the account event processor factory. Default (entered as one line): account.EventProcessorFactory = com.ibm.itim.remoteservices.ejb.reconciliation.AccountEventProcessorFactory</td>
</tr>
<tr>
<td>person.EventProcessorFactory</td>
<td>Do not modify this property key and value. Specifies the built-in Java class for the person event processor factory. Default (entered as one line): person.EventProcessorFactory = com.ibm.itim.remoteservices.ejb.reconciliation.PersonEventProcessorFactory</td>
</tr>
</tbody>
</table>

### Reconciliation processing

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>account.ReconEntryHandlerFactory</td>
<td>Do not modify this property key and value. Specifies the built-in Java class for the account entry handler factory. Default (entered as one line): account.ReconEntryHandlerFactory = com.ibm.itim.remoteservices.ejb.mediation.AccountEntryHandlerFactory</td>
</tr>
<tr>
<td>person.ReconEntryHandlerFactory</td>
<td></td>
</tr>
</tbody>
</table>
Do not modify this property key and value.

Specifies the built-in Java class for the person entry handler factory.

Default (entered as one line):

```
```
**Shared Secret Hashing**

The following property configures the level of protection of the shared secret code.

In the context of personal information management, the shared secret is the password code used by an account owner to retrieve a password for a new account. A shared secret is required if the Tivoli Identity Manager server system generates the initial password for the new account.

<table>
<thead>
<tr>
<th>enrole.sharedsecret.hashed</th>
</tr>
</thead>
<tbody>
<tr>
<td>This property key and value should be changed only by a qualified administrator.</td>
</tr>
<tr>
<td>Specifies whether the shared secret code is hashed (secure) or not hashed (not secure).</td>
</tr>
<tr>
<td>Values include:</td>
</tr>
<tr>
<td>• true – store the shared secret as hashed</td>
</tr>
<tr>
<td>• false – store the shared secret as not hashed</td>
</tr>
<tr>
<td>Example (default):</td>
</tr>
<tr>
<td>enrole.sharedsecret.hashed = false</td>
</tr>
</tbody>
</table>
## SSL Two-way Authentication Properties

These properties are no longer valid and are not used.

<table>
<thead>
<tr>
<th>Property</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.ibm.daml.jndi.DAMLContext.CLIENT_CERT</td>
<td>NOT USED</td>
</tr>
<tr>
<td>com.ibm.daml.jndi.DAMLContext.CLIENT_CERT_KEY</td>
<td>NOT USED</td>
</tr>
<tr>
<td>com.ibm.daml.jndi.DAMLContext.CLIENT_CERT_KEY_PASSPHASE</td>
<td>NOT USED</td>
</tr>
</tbody>
</table>
Request Management UI Configuration

The following property configures the amount of "Result Detail" information displayed for a failed request.

<table>
<thead>
<tr>
<th>webclient.request.maxResultDetailLines</th>
</tr>
</thead>
<tbody>
<tr>
<td>This property is used by the request management function of the Tivoli Identity Manager GUI to display the error information captured by the audit log of a failed request. The value specifies the number of lines to display for the &quot;Result Detail&quot; section of the audit log &quot;Request Details&quot;.</td>
</tr>
<tr>
<td>Home &gt; Completed Requests &gt; Request Details &gt; Audit Log &gt; Request Details &gt; Result Detail</td>
</tr>
<tr>
<td>Example (default): webclient.request.maxResultDetailLines = 20</td>
</tr>
</tbody>
</table>
Application Client Request Configuration

The following property configures the lifetime, or timeout, value for the authentication token used by the Tivoli Identity Manager application API to allow third-party applications to communicate with the Tivoli Identity Manager Server.

<table>
<thead>
<tr>
<th>authTokenTimeout</th>
</tr>
</thead>
<tbody>
<tr>
<td>This property is used by the Tivoli Identity Manager application API to specify the lifetime, or timeout, value (in hours) for the authentication token used for communication between third-party applications and the Tivoli Identity Manager Server.</td>
</tr>
<tr>
<td>A value of -1 indicates that there is no timeout for the authentication token.</td>
</tr>
<tr>
<td>Example (default):</td>
</tr>
<tr>
<td>authTokenTimeout = 48</td>
</tr>
</tbody>
</table>
Chapter 11. Configuring Supplemental Properties

This chapter provides detailed information about the property keys and values contained in the Tivoli Identity Manager supplemental configuration files.

Section topics:
• “Understanding Properties Files” on page 128

Properties file references:
• “adhocreporting.properties” on page 129
• “crystal.properties” on page 132
• “DataBaseFunctions.conf” on page 133
• “enRoleAuthentication.properties” on page 134
• “enRoleDatabase.properties” on page 137
• “enRoleLDAPConnection.properties” on page 141
• “enRoleLogging.properties” on page 144
• “enRoleMail.properties” on page 147
• “enrolepolicies.properties” on page 149
• “enroleworkflow.properties” on page 151
• “fesiextensions.properties” on page 153
• “UI.properties” on page 155
• “CustomLabels.properties” on page 158
• “Additional property files” on page 158
Understanding Properties Files

Java properties files define attributes that allow customizing and control of the Java software. Standard system properties files and custom properties files are used to configure user preferences and user customizations. A Java properties file defines the values of named resources that can specify program options such as database access information, environment settings, and special features and functionality.

A properties file defines named resources using a property key and value pair format:

\[ \text{property-key-name} = \text{value} \]

The property-key-name is an identifier for the resource. The value is the name of the actual Java object that provides the resource.

Tivoli Identity Manager uses a number of properties files to control the functionality of the program and to allow user customization of special features.
The adhocreporting.properties file supports the custom reporting module. The file contains:

- Configuration properties for custom reporting
- Display label attributes for globalization of the Tivoli Identity Manager GUI

Refer to the "Reports" chapter of the IBM Tivoli Identity Manager Policy and Organization Administration Guide for information on custom reports configuration.

Configuration properties for custom reporting are described in the following table:

<table>
<thead>
<tr>
<th>Report Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>reportPageSize</td>
</tr>
<tr>
<td>This property indicates the number of rows displayed on each page of a PDF report. The maximum number of rows on a page must not exceed 45.</td>
</tr>
<tr>
<td>Example (default, maximum):</td>
</tr>
<tr>
<td>reportPageSize = 45</td>
</tr>
</tbody>
</table>

| reportColWidth                         |
| This property indicates the width, in centimeters (cm), of the report column in a PDF report output. You can adjust the size of all columns by modifying this value. |
| Note: 2.54 cm = 1 inch                 |
| Example (default):                     |
| reportColWidth = 20                    |

<table>
<thead>
<tr>
<th>Changelog-based Data Synchronization</th>
</tr>
</thead>
<tbody>
<tr>
<td>changelogEnabled</td>
</tr>
<tr>
<td>This property is required if you configure the Incremental Data Synchronizer. Values include:</td>
</tr>
<tr>
<td>• true – Incremental Data Synchronizer configured</td>
</tr>
<tr>
<td>• false – Incremental Data Synchronizer not configured</td>
</tr>
<tr>
<td>Example (default):</td>
</tr>
<tr>
<td>changelogEnabled = false</td>
</tr>
</tbody>
</table>

| changelogBaseDN                        |
| This property contains the DN in the directory where the changelog is configured. |
| Example (default):                     |
| changelogBaseDN = cn=changelog         |

| changeLogFetchSize                     |
This property specifies the number of changelogs to be fetched at one time from the directory server.

A value of 0, or a negative value, results in no fetch restriction. Fetch restriction is useful when the directory server is not to be heavily loaded for a period of time. For example, retrieving 100,000 change log entries at a time can delay the directory server response time for a few minutes.

Example (default):
changeLogFetchSize = 200

maximumChangeLogsToSynchronize

This property specifies the maximum number of changelogs to be synchronized in a single invocation of the Incremental Data Synchronizer.

Consider the available system memory and CPU utilization required for other processes in the host when setting this property. If the value is set to zero, or a negative value, the Incremental Data Synchronizer will synchronize all changelog entries.

Example (default):
maximumChangeLogsToSynchronize = 10000

changeLogsToAnalyzeBeforeSynchronization

This property specifies the number of fetched changelog entries to be analyzed before all analyzed entries are synchronized to the database.

For example, consider the following values:
changeLogFetchSize = 500
changeLogsToAnalyzeBeforeSynchronization = 20000
maximumChangeLogsToSynchronize = 100000

500 changelog entries are considered one batch. After 20,000 changelog entries (40 batches) are analyzed, data synchronization occurs. This process repeats until 100,000 entries have been analyzed (5 synchronizations).

Setting this value to 0, or a negative value, results in synchronization of all fetched changelog entries.

Example (default):
changeLogsToAnalyzeBeforeSynchronization = 5000

ACI Enforcement on Report Data Generated

availableForNonAdministrators

This property can be used to disable ACI-related information from synchronizing during data synchronization.

Set this value to "true" if you want non-administration ITIM users to execute reports.

If the value is set to "false", all functionalities related to non-administrator execution of reports, such as ACI data synchronization and setting report ACIs on custom reports, will be disabled.

Example (default):
availableForNonAdministrators = true
**Incremental Schema Enforcement using Incremental Data Synchronizer**

| enableDeltaSchemaEnforcer | This property enables or disables synchronization of any schema changes in custom reporting. Schema changes include new mappings created or existing mappings removed using the Schema Designer. When set to "true", the Incremental Data Synchronizer will add/modify/delete data for the attributes which are mapped (changed) in the Schema Designer since the last full data synchronization was run. When set to "false", the Incremental Data Synchronizer will not synchronize the attributes which are mapped (changed) since the last full data synchronization was run. Example (default): enableDeltaSchemaEnforcer = false |

**Configurations for Full Data Synchronization**

| createIndex | This property enables and disables the creation of database indices for frequently used database columns. Substantial improvement in performance of report generation is possible by enabling this option. To enable creation of indices, set this property to "true". Valid values for this property are:  
  • **true** – Create indices for metadata columns used by custom reporting. Please note that enabling this value might increase the data synchronization time.  
  • **false** – Do not create indices during data synchronization. Disabling creation of indices might increase the time required for report generation. Example (default): createIndex = true |

| sqlBatchSize | This property indicates the size of batch updates processed during data synchronization. Substantial improvement in performance is possible by setting this value to a higher number. However, this value is affected by the specific database settings for the transaction log file size (a database property) and setting the value too high might cause data synchronization to fail. Always use the recommended default value of 50 to avoid data synchronization failure. Example (default): sqlBatchSize = 50 A value of 0, or a negative value, results in the processing of all SQL updates as a single batch. |
The `crystal.properties` file stores the global properties for the Crystal Reports plugin of the custom reporting module. Refer to the "Configuring Crystal Reports" chapter in the *IBM Tivoli Identity Manager Configuration Guide* (this document) for information on Crystal Reports configuration.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>crystalras</code></td>
<td>Specifies the server name or IP address of the machine where the Crystal Report Application Server is installed</td>
<td><code>crystalras = 999.999.999.999</code></td>
</tr>
<tr>
<td><code>dsn</code></td>
<td>Crystal functionality requires that a DSN be created on the RAS server machine. This property specifies the name of the DSN.</td>
<td><code>dsn = &lt;string&gt;</code></td>
</tr>
<tr>
<td><code>database</code></td>
<td>Specifies the name of the database to which the DSN points to.</td>
<td><code>database = &lt;string&gt;</code></td>
</tr>
</tbody>
</table>
The custom reporting feature of Tivoli Identity Manager allows you to use database functions when designing custom report templates. Database functions can be made available for use with the Report Designer component of the Tivoli Identity Manager GUI by defining the functions in the `DataBaseFunctions.conf` file.

Pre-defined database functions use the following format in `DataBaseFunctions.conf`:

```
<function_name> - <number_of_arguments>
```

Database users can also create and define functions for their custom use. Custom functions are referred to as "User-Defined functions" in Microsoft SQL and IBM DB2. Functions created as "Stored Procedures" in DB2 can also be used for reporting. Functions must be created using the database utilities provided by the respective database vendor.

User-defined database functions use the following format in `DataBaseFunctions.conf`:

```
user:<function_name> - <number_of_arguments>
```

Only functions with a single argument are supported in the Tivoli Identity Manager Report Designer. For advanced usage, users are requested to use the Crystal Reports Designer Tool.

```
<table>
<thead>
<tr>
<th>Upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Converts the argument to upper case.</td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td>Upper - 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lower</th>
</tr>
</thead>
<tbody>
<tr>
<td>Converts the argument to lower case.</td>
</tr>
<tr>
<td>Lower - 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>user:GetDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-defined. Get the Distinguished Name.</td>
</tr>
<tr>
<td>user:GetDN - 1</td>
</tr>
</tbody>
</table>
```
**enRoleAuthentication.properties**

The `enRoleAuthentication` file specifies the type of method used by the Tivoli Identity Manager system to authenticate users and identifies the Java object that provides the specified authentication mechanism. Additionally, the file specifies objects that support Tivoli Access Manager WebSEAL single sign-on and administration of Tivoli Identity Manager access to managed remote services.

Authentication properties are specified using a property key and value pair format:

```
property-key-name = value
```

The `property-key-name` is an identifier for the authentication mechanism or resource. The `value` is the name of the Java object that provides the authentication service, expressed also as a key and value pair:

```
factory = value
```

The `factory` key name represents a special category for authentication support within the Tivoli Identity Manager software. The `value` is the actual name of the Java object.

For example (entered as one line):

```
enrole.authentication.provider.service =
   factory = com.ibm.enrole.authentication.service.
   ServiceAuthenticationProviderFactory
```

### Authentication method

- `enrole.authentication.requiredCredentials = {simple|certificate}`
  - Specifies either **simple** (user name and password) or **certificate** (digital certificate) as the required authentication method for users logging into the Tivoli Identity Manager system.
  - The system default is **simple**.
  - For example:
    ```
enrole.authentication.requiredCredentials = simple
    ```
  - You can alternatively specify a custom authentication mechanism. See [“Configuring a custom authentication mechanism” on page 135](#).

### Authentication providers (factories)

- `enrole.authentication.provider.simple`
  - Specifies the Java object that handles authentication with user name and password.
  - For example (entered as one line):
    ```
enrole.authentication.provider.simple =
   factory = com.ibm.enrole.authentication.simple.
   SimpleAuthenticationProviderFactory
```

- `enrole.authentication.provider.certificate`
  - Specifies the Java object that handles authentication with digital certificates.
  - For example (entered as one line):
    ```
enrole.authentication.provider.certificate =
   factory = com.ibm.enrole.authentication.certificate.
   CertificateAuthenticationProviderFactory
```
## Authentication service provider

**enrole.authentication.provider.service**

Specifies the Java object that transparently handles Tivoli Identity Manager access to managed remote services and to manage changes in the accounts to these remote services.

These changes include addition, deletion, suspension, restoration, and modification of accounts on the remote service. When logged into Tivoli Identity Manager, you can change the login and password information for an account on the managed remote service.

The `ServiceAuthenticationProviderFactory` mechanism works with the agent for a given remote service and processes the changed information.

For example (entered as one line):

```java
enrole.authentication.provider.service =
    factory = com.ibm.enrole.authentication.service.
ServiceAuthenticationProviderFactory
```

## WebSEAL single sign-on

**enrole.authentication.provider.webseal**

Specifies the Java object that allows single sign-on in a WebSEAL environment.

For example (entered as one line):

```java
enrole.authentication.provider.webseal =
    factory = com.ibm.enrole.authentication.webseal.WebsealProviderFactory
```

See ["Configuring Single Sign-on with WebSEAL" on page 40](#)

**enrole.authentication.idsEqual**

Indicates the appropriate algorithm for mapping the Tivoli Access Manager user ID to a Tivoli Identity Manager user ID.

If the Tivoli Access Manager user ID is the same as the Tivoli Identity Manager user ID (default setting):

```java
enrole.authentication.idsEqual = true
```

If the Tivoli Access Manager user ID is NOT the same as the Tivoli Identity Manager user ID:

```java
enrole.authentication.idsEqual = false
```

An internal identity mapping algorithm is used to ensure the success of the single sign-on operation.

See ["Configuring Single Sign-on with WebSEAL" on page 40](#)

---

**Configuring a custom authentication mechanism**

As an alternative to the built-in user name/password and certificate authentication methods provided by Tivoli Identity Manager, you can use the `enrole.authentication.requiredCredentials` property key to specify a custom Java object that invokes a custom authentication mechanism.

For example, you might require a single sign-on to Tivoli Identity Manager from a signed-on session with a portal server.
Create the custom authentication Java object with the assistance of Tivoli Customer Support and enter this object as the value to the `enrole.authentication.requiredCredentials` property key.

The `enRoleAuthentication.properties` file allows Tivoli Identity Manager to use only one type of authentication method. You cannot configure multiple concurrent authentication mechanisms.
The enRoleDatabase.properties file specifies attributes that support the relational database used by the Tivoli Identity Manager workflow engine. Tivoli Identity Manager supports three database types:

- DB2
- Oracle
- MS SQL Server

The property key values contained in this file are synchronized with values in the appropriate application server configuration file. Most values in this file are supplied during initial installation of Tivoli Identity Manager and the configuration of the database. You can make subsequent changes to some values. However, you must use the runConfig utility to synchronize the property file values with the values in the application server configuration file.

Tivoli Identity Manager uses Java Database Connectivity (JDBC) to access the relational database. JDBC technology is an API that lets you access virtually any tabular data source from the Java programming language.

<table>
<thead>
<tr>
<th>Database information</th>
</tr>
</thead>
<tbody>
<tr>
<td>database.db.type</td>
</tr>
</tbody>
</table>
| Do not modify this property key. The value is supplied during initial installation of Tivoli Identity Manager. Specifies the database type used by the Tivoli Identity Manager workflow engine. Appropriate values include:
  • DB2
  • Oracle
  • MS SQL Server

Example (DB2):
  database.db.type = DB2

<table>
<thead>
<tr>
<th>database.db.server</th>
</tr>
</thead>
</table>
| This value is supplied during initial installation of Tivoli Identity Manager and configuration of the database. Specifies the name or local alias name of the remote database.

To change this value for a new database, use the database configuration utility to set up the database. The database configuration utility supplies the new database name to this properties file.

To change this value for another existing database, use the runConfig utility to supply the new database name to this properties file.

Example:
  database.db.server =itimdb
### database.db.owner

Do not modify this property key. The value is built-in to the system.

Specifies the name of the database schema owner for Tivoli Identity Manager.

Example (built-in):
```
database.db.owner = enrole
```

### database.db.user

Do not modify this property key. The value is built-in to the system.

Specifies a default database user for Tivoli Identity Manager.

Example (built-in):
```
database.db.user = enrole
```

### database.db.password

Do not modify this property key. The value is supplied during database configuration.

Specifies the password for the database user.

Encryption of this value is specified by the `enrole.password.database.encrypted` property in `enrole.properties`.

The password value appears as clear text unless an encryption setting has been activated using the `runConfig` utility.

Example (clear text):
```
database.db.password = secret
```

### Connection pool properties

#### database.jdbc.connectionPool.initialCapacity

Do not manually edit this file to modify this property key value. Use the `runConfig` utility to change this value.

Specifies the initial number of physical database connections to create for the connection pool. This value must be less than or equal to the `maxCapacity` value.

Example (default):
```
database.jdbc.connectionPool.initialCapacity = 5
```

#### database.jdbc.connectionPool.maxCapacity

Do not manually edit this file to modify this property key value. Use the `runConfig` utility to change this value.

Specifies the maximum number of physical database connections that can be created. This value limit addresses system performance tuning requirements.

Example (default):
```
database.jdbc.connectionPool.maxCapacity = 50
```
### `database.jdbc.connectionPool.capacityIncrement`

You can manually edit this file to modify this property key value. Additionally, you must use the `runConfig` utility to synchronize the changes with the application server configuration file.

Specifies the number of connections created any time the connection pool is expanded (incremented) to accommodate increasing demand.

Example (default):

```properties
database.jdbc.connectionPool.capacityIncrement = 1
```

### `database.jdbc.connectionPool.loginDelaySecs`

You can manually edit this file to modify this property key value. Additionally, you must use the `runConfig` utility to synchronize the changes with the application server configuration file.

Specifies the delay period (in seconds) between the creation of each database connection.

Example (default):

```properties
database.jdbc.connectionPool.loginDelaySecs = 1
```

### `database.jdbc.connectionPool.ShrinkingEnabled`

You can manually edit this file to modify this property key value. Additionally, you must use the `runConfig` utility to synchronize the changes with the application server configuration file.

Specifies whether or not the connection pool can shrink back to its initialCapacity value when it is detected that additional connections created during increased traffic are no longer being used. Values include "true" or "false".

Example (default):

```properties
database.jdbc.connectionPool.ShrinkingEnabled = true
```

### `database.jdbc.connectionPool.ShrinkPeriodMinutes`

You can manually edit this file to modify this property key value. Additionally, you must use the `runConfig` utility to synchronize the changes with the application server configuration file.

Specifies the number of minutes to wait before shrinking a connection pool that has incrementally increased to meet demand. `ShrinkingEnabled` must be set to "true" for a connection pool to shrink.

Example (default):

```properties
database.jdbc.connectionPool.ShrinkPeriodMinutes = 15
```

### `database.jdbc.connectionPool.Targets`

Do not modify this property key value. The value is supplied during initial installation of Tivoli Identity Manager.

Specifies the target in the current domain on which this connection pool can be deployed. The value typically is the application server name or the cluster name.

Example:

```properties
database.jdbc.connectionPool.Targets = myserver
```
### database.jdbc.connectionPool.testTableName

Do not modify this property key value.

Specifies the name of the table used when testing a physical database connection. Each database type requires its own value:

- DB2 – "nextvalue"
- Oracle – "dual"
- MS SQL Server – "nextvalue"

Example (Oracle):

```java
database.jdbc.connectionPool.testTableName = dual
```

### database.jdbc.connectionPool.refreshMinutes

Do not modify this property key value.

Specifies the number of minutes between database connection tests.

Example (default):

```java
database.jdbc.connectionPool.refreshMinutes = 5
```

### JDBC driver

### database.jdbc.driverURL

Do not remove or modify this property key and value.

Specifies the JDBC driver URL.

Example (DB2):

```java
database.jdbc.driverUrl = jdbc:db2:itimdb
```

### database.jdbc.driver

Do not remove or modify this property key and value.

Specifies the JDBC driver name.

Example (DB2):

```java
database.jdbc.driver = COM.ibm.db2.jdbc.app.DB2Driver
```
The enRoleLDAPConnections.properties file provides standard configuration settings that allow successful communication between Tivoli Identity Manager and the LDAP directory server.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.naming.factory.initial</td>
<td>Specifies the built-in Java class file that provides the communication interface between Tivoli Identity Manager and the LDAP directory server. The Java Naming and Directory Interface (JNDI) protocol is used.</td>
</tr>
<tr>
<td>Do not modify this property key and value.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.naming.provider.url</td>
<td>Specifies the location (URL) of the LDAP directory server. The LDAP server can be located on:</td>
</tr>
<tr>
<td>• The local Tivoli Identity Manager machine</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use &quot;localhost&quot;.</td>
</tr>
<tr>
<td>• A remote machine</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use short or fully qualified host name, or IP address.</td>
</tr>
<tr>
<td>The value for this key is initially configured during Tivoli Identity Manager installation. You can also provide this value using the ldapconfig utility or runConfig utility.</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>java.naming.provider.url = ldap://localhost:389</td>
<td></td>
</tr>
<tr>
<td>LDAP context: Context.PROVIDER_URL</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.naming.security.principal</td>
<td>Specifies the Distinguished Name (DN) of the LDAP administration account on the LDAP directory server.</td>
</tr>
<tr>
<td>The value for this key is initially configured during Tivoli Identity Manager installation. You can also provide this value using the ldapconfig utility or runConfig utility.</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>java.naming.security.principal = cn = root</td>
<td></td>
</tr>
<tr>
<td>LDAP context: ContextSECURITY_PRINCIPAL</td>
<td></td>
</tr>
</tbody>
</table>
### java.naming.security.credentials

Specifies the password for the LDAP administration account on the LDAP directory server.

The value for this key is initially configured during Tivoli Identity Manager installation. You can also provide this value using the `ldapconfig` utility or `runConfig` utility.

Encryption of this value is specified by the `enrole.password.ldap.encrypted` property in `enrole.properties`.

The encryption type is initially configured during Tivoli Identity Manager installation using the Encryption Setting check box.

Example:
```
java.naming.security.credentials = ibmldap
```

LDAP context: `Context.SECURITY_CREDENTIALS`

### java.naming.security.protocol

This property key and value are not enabled for this release of Tivoli Identity Manager.

Do not modify this property key and value.

Specifies the protocol used for communication between Tivoli Identity Manager and the LDAP directory server.

LDAP context: `Context.SECURITY_PROTOCOL`

### java.naming.security.authentication

Do not modify this property key and value.

Specifies the authentication type used by the LDAP directory server. Valid types include:

- **none** (anonymous: user becomes a member of an unauthenticated group)
- **simple** (user name and password required)
- **strong** (place-holder for stronger authentication mechanism)

Example:
```
java.naming.security.authentication = simple
```

LDAP context: `Context.SECURITY_AUTHENTICATION`
java.naming.referral

Do not modify this property key and value.

In a scenario where multiple LDAP directory servers are linked together in the Tivoli Identity Manager environment, specifies whether links are to be used when a referral is indicated to complete a request for LDAP information.

Valid values include:
  • follow (accept the use of links)
  • ignore (do not use links)
  • throw (do not use links and return an error message)

Example:
java.naming.referral = follow
LDAP context: Context.REFERRAL

java.naming.batchsize

Do not modify this property key and value.

A JNDI property that specifies the number of data elements returned at one time during a request (query) to the LDAP directory server. A larger number reduces the number of LDAP fetches, resulting in improved performance.

A "0" value blocks any control by the client (Tivoli Identity Manager) until all requested elements are returned.

Example:
java.naming.batchsize = 100
LDAP context: Context.BATCHSIZE

java.naming.ldap.attributes.binary

Do not modify this property key and value.

Specifies Tivoli Identity Manager attributes that are treated as binary data type. Multiple attribute values are separated by a single space.

Example:
java.naming.ldap.attributes.binary = erPassword erHistoricalPassword
LDAP context: attribute.binary
enRoleLogging.properties

The enRoleLogging.properties file specifies attributes that govern the operation of the log4j logging and tracing API that is bundled with Tivoli Identity Manager.

log4j is a popular logging package for Java, distributed under an open source Apache software license and certified by the open source initiative. log4j allows you to log messages according to message type and priority, and to control at runtime how these messages are formatted and where they are reported.

The log4j package includes excellent documentation. Please refer to this information for a full understanding of the log4j functionality.

<table>
<thead>
<tr>
<th>Enable/disable traceExceptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.logprovider.traceexceptions</td>
</tr>
</tbody>
</table>

A Tivoli Identity Manager attribute (not a log4j attribute). Specifies whether or not additional detailed error information (call stack) is added to the message logs. Values include:

- true (log detailed information)
- false (do not log detailed information)

Default is "true".

Example:

enrole.logprovider.traceexceptions = true

Category hierarchy setup

Root category:

log4j.rootCategory


log4j categories are named entities that allow you to associate components of the system with a logging priority. The root category represents all system components (all components are children of the root parent).

The log4j.rootCategory property key specifies a system default logging priority and defines appender (output destination) names that are used to specify output destination types.

Priority hierarchy:

1. FATAL
2. ERROR
3. WARN
4. INFO
5. DEBUG

For example, a priority of INFO specifies that all INFO, WARN, ERROR, and FATAL messages are logged.

Example (specifying a WARN logging priority and an appender entity named "Logger"):

log4j.rootCategory = WARN, Logger
Component categories:

log4j.category.com.ibm.itim.apps
log4j.category.com.ibm.itim.authentication
log4j.category.com.ibm.itim.authorization
log4j.category.com.ibm.itim.common
log4j.category.com.ibm.itim.fesiextensions
log4j.category.com.ibm.itim.logging
log4j.category.com.ibm.itim.mail
log4j.category.com.ibm.itim.messaging
log4j.category.com.ibm.itim.migration
log4j.category.com.ibm.itim.dataservices.model
log4j.category.com.ibm.itim.passworddelivery
log4j.category.com.ibm.itim.policy
log4j.category.com.ibm.itim.remoteservices
log4j.category.com.ibm.itim.report
log4j.category.com.ibm.itim.security
log4j.category.com.ibm.itim.scheduling
log4j.category.com.ibm.itim.systemConfig
log4j.category.com.ibm.itim.util
log4j.category.com.ibm.itim.webclient
log4j.category.com.ibm.itim.workflow

These categories represent standard Tivoli Identity Manager components. You can configure each component individually to override the root category logging priority. Uncomment the line to enable the component logging configuration.

Example:

log4j.category.com.ibm.itim.policy = INFO

In the example above, the logging priority of INFO adds additional message logging for the itim.policy component (the example root category is set only to WARN).

Log file output destination (appender)

log4j.appender.appender-identifier

An appender specifies an output destination type for the log file. Types include:

- Single file
- Roll-over file
- Console (output to screen)
- NT Event Loggers

Example (the appender entity "Logger" is set to the following Java class that processes a roll-over file type):

log4j.appender.Logger = org.apache.log4j.RollingFileAppender

Example properties for RollingFileAppender:

log4j.appender.Logger.File = c:/temp/itim.log
log4j.appender.Logger.MaxFileSize = 2MB
log4j.appender.Logger.MaxBackupIndex = 10

For more detailed information, refer to the "Message Logging" chapter of the IBM Tivoli Identity Manager Problem Determination Guide.
Log file layout and conversion pattern specification

<table>
<thead>
<tr>
<th>log4j.appender.appender-identifier.layout</th>
</tr>
</thead>
<tbody>
<tr>
<td>The layout specification is responsible for formatting the request appropriately. A Java class that performs the layout operation is identified and a conversion pattern is specified.</td>
</tr>
<tr>
<td>Example (specifies layout pattern Java class used by the &quot;Logger&quot; appender entity): log4j.appender.Logger.layout = org.apache.log4j.PatternLayout</td>
</tr>
<tr>
<td>Example conversion pattern for PatternLayout: log4j.appender.Logger.layout.ConversionPattern = [%d:%t]&lt;%p:%c&gt;%m%n</td>
</tr>
<tr>
<td>Above conversion pattern interpretation: [date:thread-id]<a href="">priority-level:category</a>message line break</td>
</tr>
</tbody>
</table>
enRoleMail.properties

The enRoleMail.properties file contains attributes that specify the mail transport protocol used by the built-in JavaMail API and other Tivoli Identity Manager application-specific properties.

You must provide the values for the application-specific property keys.

Default values are provided for the JavaMail-specific property keys (including the default mail provider and protocol). If you change the default values for the JavaMail-specific property keys, you must provide your own testing and verification of your custom protocol and implementation.

Refer to the following URL for additional usage and provider information:

http://java.sun.com/products/javamail/

<table>
<thead>
<tr>
<th>Mail attributes specific to the Tivoli Identity Manager application</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>mail.from</strong></td>
</tr>
<tr>
<td>Required.</td>
</tr>
<tr>
<td>This value is initially provided during Tivoli Identity Manager installation. You can also provide the value using the runConfig utility.</td>
</tr>
<tr>
<td>Specifies the return e-mail address of the current user.</td>
</tr>
<tr>
<td>Example (user-provided):</td>
</tr>
<tr>
<td>mail.from = <a href="mailto:admin@us.ibm.com">admin@us.ibm.com</a></td>
</tr>
<tr>
<td><strong>mail.baseurl</strong></td>
</tr>
<tr>
<td>Required.</td>
</tr>
<tr>
<td>Specifies the base URL that is used to construct the login URL in e-mails sent to new Tivoli Identity Manager users.</td>
</tr>
<tr>
<td>This value is initially provided during Tivoli Identity Manager installation. You can also provide the value using the runConfig utility.</td>
</tr>
<tr>
<td>See also &quot;Web Server Information&quot; on page 11</td>
</tr>
<tr>
<td>Example (user-provided):</td>
</tr>
<tr>
<td>mail.baseurl = <a href="http://111.222.333.444:80">http://111.222.333.444:80</a></td>
</tr>
<tr>
<td><strong>mail.title</strong></td>
</tr>
<tr>
<td>Required.</td>
</tr>
<tr>
<td>You must edit this properties file directly to provide the value to this property key.</td>
</tr>
<tr>
<td>Specifies the text string to use in the title banner of e-mail messages. Default string is &quot;ITIM notification&quot;.</td>
</tr>
<tr>
<td>Example (user-provided):</td>
</tr>
<tr>
<td>mail.title = ITIM notification</td>
</tr>
<tr>
<td><strong>Mail attributes specific to the built-in Java mail service</strong></td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>mail.host</strong></td>
</tr>
<tr>
<td>Required.</td>
</tr>
<tr>
<td>The value is initially provided during Tivoli Identity Manager installation. You can also provide this value using the runConfig utility.</td>
</tr>
<tr>
<td>Specifies the IP address of the machine where the mail server is located.</td>
</tr>
<tr>
<td>Example (user-provided): mail.host = 111.222.333.444</td>
</tr>
<tr>
<td><strong>mail.protocol.host</strong></td>
</tr>
<tr>
<td>Specifies the IP address of the protocol-specific default mail server. This property key overrides the mail.host property key.</td>
</tr>
<tr>
<td>By default, this property is not required and no value is provided.</td>
</tr>
<tr>
<td><strong>mail.transport.protocol</strong></td>
</tr>
<tr>
<td>Specifies the default transport protocol (Sun SMTP transport).</td>
</tr>
<tr>
<td>Example (default): mail.transport.protocol = SMTP</td>
</tr>
<tr>
<td><strong>mail.protocol.class</strong></td>
</tr>
<tr>
<td>Specifies the Java class implementation of the default Sun SMTP mail protocol.</td>
</tr>
<tr>
<td>Example (default): mail.SMTP.class = com.sun.mail.smtp.SMTPTransport</td>
</tr>
<tr>
<td><strong>mail.store.protocol</strong></td>
</tr>
<tr>
<td>Specifies the default message access protocol.</td>
</tr>
<tr>
<td>By default, this property is not required and no value is provided.</td>
</tr>
<tr>
<td><strong>mail.user</strong></td>
</tr>
<tr>
<td>Specifies the user name used during authentication when connecting to a mail server.</td>
</tr>
<tr>
<td>By default, this property is not required and no value is provided. In the Tivoli Identity Manager environment, the mail server is located within firewall boundaries, rendering this level of authentication unnecessary.</td>
</tr>
<tr>
<td><strong>mail.protocol.user</strong></td>
</tr>
<tr>
<td>Specifies the protocol-specific user name used during authentication when connecting to a mail server. This property key overrides the mail.user property key.</td>
</tr>
<tr>
<td>By default, this property is not required and no value is provided.</td>
</tr>
</tbody>
</table>
enrolepolicies.properties

The enrolepolicies.properties file provides standard and custom settings that support functionality of the Tivoli Identity Manager provisioning policy. Functionality supported by this properties file include:

- Specifying Java classes to process provisioning policy conflicts using join directives
- Specifying default and non-default join directive caching timeouts
- Declaring policy attributes to be ignored during policy compliance validation

A join directive is a set of rules used to determine how attributes are handled when more than one provisioning policies conflict. Join directives use logical constructs to resolve conflicts. Examples include combining all policy attributes (union), using only common attributes (intersection), and resolving conflicts using Boolean "AND"/"OR" logic.

There are a total of 12 types of join directives available for use in Tivoli Identity Manager. Provisioning policy join directives take effect when there is more than one provisioning policy defined for the same user (or group of users) for the same target service, service instance, or service type.

Additionally, custom join directives can be defined by writing a custom Java class and adding the custom property key and value to this file.

<table>
<thead>
<tr>
<th>Join directive classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>provisioning.policy.join.PrecedenceSequence = com.ibm.enrole.policy.join.PrecedenceSequence</td>
</tr>
<tr>
<td>provisioning.policy.join.Textual = com.ibm.enrole.policy.join.Textual</td>
</tr>
<tr>
<td>provisioning.policy.join.Multivalued = com.ibm.enrole.policy.join.Multivalued</td>
</tr>
</tbody>
</table>

Do not modify these property keys and values. Each property key specifies a Java class that can be used to process a join directive logic operation required to resolve a provisioning policy conflict.

<table>
<thead>
<tr>
<th>Append separator characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>provisioning.policy.join.Textual.AppendSeparator</td>
</tr>
</tbody>
</table>

Specifies the character type used by the Textual join directive Java class to separate individual values of a multi-value attribute.

Example:
provisioning.policy.join.Textual.AppendSeparator = <<<<<>
### Join directive cache timeouts

**provisioning.policy.join.defaultCacheTimeout**

Specifies the timeout interval between refreshes of the cache that stores default join directive cache values. The timeout value is recorded in seconds.

(Default = 86400 seconds = 24 hours).

Example:

```plaintext
provisioning.policy.join.defaultCacheTimeout = 86400
```

**provisioning.policy.join.overridingCacheTimeout**

Specifies the timeout interval between refreshes of the cache that stores non-default join directive values. The timeout value is recorded in seconds.

(Default = 300 seconds = 5 minutes).

Example:

```plaintext
provisioning.policy.join.overridingCacheTimeout = 300
```

### Account attributes ignored by policy compliance validation

**Excluded generic attributes:**

- `nonvalidateable.attribute.eraccountcompliance`
- `nonvalidateable.attribute.eracl`
- `nonvalidateable.attribute.eraccountstatus`
- `nonvalidateable.attribute.erauthorizationowner`
- `nonvalidateable.attribute.erglobalid`
- `nonvalidateable.attribute.erhistoricalpassword`
- `nonvalidateable.attribute.erisdeleted`
- `nonvalidateable.attribute.erlastmodifiedtime`
- `nonvalidateable.attribute.ernumlogons`
- `nonvalidateable.attribute.erparent`
- `nonvalidateable.attribute.erpassword`
- `nonvalidateable.attribute.erservice`
- `nonvalidateable.attribute.eruid`
- `nonvalidateable.attribute.objectclass`
- `nonvalidateable.attribute.owner`

**Excluded Windows NT attributes:**

- `nonvalidateable.attribute.erntpasswordexpired`
- `nonvalidateable.attribute.ernuserbadpwdcount`
- `nonvalidateable.attribute.ernntlockedout`

Declares account attributes that are to be ignored during policy compliance validation. This exclusion list helps reduce unnecessary overhead during compliance validation and reduces the risk of system failure caused by attributes that cannot logically be resolved during validation.
enroleworkflow.properties

The enroleworkflow.properties file specifies the XML file mappings for system-defined workflows. In Tivoli Identity Manager, a workflow is a process which specifies the flow of operations that involve business services and human interactions. A workflow design defines the manner in which a particular business logic is processed. The XML files specified in the enroleworkflow.properties file implement workflow designs.

The system workflow is identified by a unique Type ID and an associated XML file. The XML workflow files are located in the following directory:

`ITIM_HOME\data\workflow_systemprocess`

Normally, you should not remove or modify the default system workflow Type IDs and XML file values provided in this file.

If your implementation of Tivoli Identity Manager requires new or redefined system workflows to support custom business logic, you can modify this file accordingly:

- New business logic – new Type IDs and XML-formatted process definition files can be added to the file
- Modified business logic – existing default XML-formatted process definition files can be replaced by custom files

You must have the proper technical and programming resources available to perform such a modification.

<table>
<thead>
<tr>
<th>Policy enforcement workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.workflow.PS = enforcepolicyforservice.xml</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service selection management workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.workflow.SA = addserviceelectionpolicy.xml</td>
</tr>
<tr>
<td>enrole.workflow.SC = changeserviceelectionpolicy.xml</td>
</tr>
<tr>
<td>enrole.workflow.SD = removeserviceelectionpolicy.xml</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Provisioning policy management workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Add policy</td>
</tr>
<tr>
<td>enrole.workflow.PA = addpolicy.xml</td>
</tr>
<tr>
<td>#Modify policy</td>
</tr>
<tr>
<td>enrole.workflow.PC = changepolicy.xml</td>
</tr>
<tr>
<td>#Delete policy</td>
</tr>
<tr>
<td>enrole.workflow.PD = removepolicy.xml</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reconciliation workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.workflow.RC = reconciliation.xml</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Change multiple user state workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>enrole.workflow.MS = multiusersuspend.xml</td>
</tr>
<tr>
<td>enrole.workflow.MR = multiuserrestore.xml</td>
</tr>
<tr>
<td>enrole.workflow.MD = multiuserdelete.xml</td>
</tr>
</tbody>
</table>
Change multiple account state workflow

| enrole.workflow.LD | multiaccountdelete.xml |
| enrole.workflow.LS | multiaccountsuspend.xml |
| enrole.workflow.LR | multiaccountrestore.xml |
| enrole.workflow.LP | multiaccountpassword.xml |

Dynamic role workflow

#Add dynamic role
enrole.workflow.DA = adddynamicrole.xml

#Modify dynamic role
enrole.workflow.DC = changedynamicrole.xml

#Delete dynamic role
enrole.workflow.DD = removedynamicrole.xml
The `fesiextensions.properties` file defines built-in and custom FESI extensions required by Tivoli Identity Manager. FESI refers to the Free EcmaScript Interpreter, a JavaScript interpreter written in Java. The FESI interpreter reads this properties file during Tivoli Identity Manager initialization to set extensions for required Java classes.

The FESI extensions represent regions ("hooks") in the Tivoli Identity Manager software where the use of JavaScript is allowed to introduce built-in or custom business logic. FESI extensions are specified using a property key and value pair format:

```
property-key-name = value
```

The value is a fully qualified Java class file name. The `property-key-name` includes a standard prefix `fesi.extension`, a context, and (for custom classes) an identifier name (ID) representing the fully qualified Java class file. Typically the shorter unqualified class name is used as the identifier name (ID).

```
fesi.extension.context.class-ID = fully-qualified-class-name
```

The FESI system extensions used by Tivoli Identity Manager include a global context and three specific contexts.

Global context identifier:

```
Enrole
```

Specific context identifiers:

```
IdentityPolicy
HostSelection
Workflow
```

Although you must not modify the built-in system FESI extensions, you can add custom FESI extensions that may be required for any custom programs. When you add a custom FESI extension to this properties file, you must use one of the established global or specific contexts.

Indicate the fully qualified custom Java class file name as the `value` and provide a unique property key identifier name (ID) for the custom class. Examples:

```
fesi.extension.IdentityPolicy.custom-class-ID = custom-fully-qualified-class-name
fesi.extension.HostSelection.custom-class-ID = custom-fully-qualified-class-name
```

### Built-in system FESI extensions

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fesi.extension.Enrole</td>
<td>com.ibm.itim.fesiextensions.Enrole</td>
</tr>
<tr>
<td>fesi.extension.HostSelection</td>
<td>com.ibm.itim.fesiextensions.ModelSelection</td>
</tr>
<tr>
<td>fesi.extension.Workflow</td>
<td>com.ibm.itim.workflow.fesiextensions.WorkflowExtension</td>
</tr>
</tbody>
</table>

The value for each system property key is a built-in fully qualified Java class file. Do not remove or modify information in this section.
### Custom FESI extensions

<table>
<thead>
<tr>
<th>Example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>fesi.extension.enRole.custom-class-ID = custom-fully-qualified-class-name</td>
</tr>
</tbody>
</table>

You can modify the fesiextensions.properties files to include additional FESI extensions for required custom objects and methods.

The value for each custom property key is a fully qualified custom Java class file.

All property key names must be unique.
The `UI.properties` file specifies attributes that affect the operation and display of the Tivoli Identity Manager GUI. In addition, there are two sections containing attributes that do not pertain to the Tivoli Identity Manager GUI.

<table>
<thead>
<tr>
<th>Tivoli Identity Manager GUI configuration settings</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enrole.ui.errorPage.verbosity</code></td>
</tr>
<tr>
<td>Specifies whether or not to display detailed information (stack trace) with an error message: Values include:</td>
</tr>
<tr>
<td>• 0 – no detailed information displayed</td>
</tr>
<tr>
<td>• 1 – detailed information displayed</td>
</tr>
<tr>
<td>Default value is &quot;0&quot;.</td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td><code>enrole.ui.errorPage.verbosity = 0</code></td>
</tr>
<tr>
<td><code>enrole.ui.customerLogo.image</code></td>
</tr>
<tr>
<td>Specifies the file name of the graphic displayed on the right-hand side of the Tivoli Identity Manager GUI title banner. The graphic is usually a company logo. For display over the Web in a browser, the format of the file must be .gif or .jpeg. The actual graphics file must be stored in the following location:</td>
</tr>
<tr>
<td>WebSphere:</td>
</tr>
<tr>
<td><code>...WebSphere/AppServer/installedApps/domain-name/enRole.ear/app_web.war/images</code></td>
</tr>
<tr>
<td>WebLogic:</td>
</tr>
<tr>
<td><code>...bea/user_projects/domain-name/applications/enrole/images</code></td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td><code>enrole.ui.customerLogo.image = ibm_banner.gif</code></td>
</tr>
<tr>
<td><code>enrole.ui.customerLogo.url</code></td>
</tr>
<tr>
<td>Specifies the URL link that is activated when you click on the custom graphic image (company logo) in the right-hand side of the Tivoli Identity Manager GUI title banner.</td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td><code>enrole.ui.customerLogo.url = www.ibm.com</code></td>
</tr>
<tr>
<td><code>enrole.ui.pageSize</code></td>
</tr>
<tr>
<td>Specifies the number of list items initially displayed on the screen. If there are more items in the list, links appear at the bottom of the list view that activate continuations of the list (for example, Page 2, Page 3, Page 4).</td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td><code>enrole.ui.pageSize = 10</code></td>
</tr>
</tbody>
</table>
### enrole.ui.pageLinkMax

Specifies the number of page links that are displayed for a long item list (see enrole.ui.pageSize). If more page links are required by the list than are specified by this property key, a "Next" link is added.

Example:
```
enrole.ui.pageLinkMax = 10
```

### enrole.ui.maxSearchResults

Specifies the number of items returned for a search. The value for this property key can control possible system performance degradation when a very large return of items is encountered.

Example:
```
enrole.ui.maxSearchResults = 1000
```

---

### WfDesigner and FormDesigner applet properties

- enrole.build.version
- enrole.java.plugin
- enrole.java.plugin.classid
- enrole.java.plugin.jpi-version
- enrole.java.pluginspage
- enrole.ui.logoffURL
- enrole.ui.timeoutURL

You must not modify or remove any information in this section. These property key and value pairs provide the necessary Java applet support required by the Web browser that runs the Tivoli Identity Manager GUI.

---

### Report menu properties

### enrole.ui.reconReport.maxFileSize

The Reconciliation report can be generated in the following formats:
- PDF
- HTML
- CVS (comma delimited file)

This property key and value specifies the limit of the report size (in bytes) for the PDF output option to be available. If the report size is greater than .5 MB, the PDF option is no longer available as an output option.

Example:
```
enrole.ui.reconReport.maxFileSize = 500000
```

### enrole.ui.accountReport.maxPeopleInReport

Specifies the maximum number of people to include in the Account report.

Example:
```
enrole.ui.accountReport.maxPeopleInReport = 500
```
<table>
<thead>
<tr>
<th><strong>enrole.ui.report.maxRecordsInReport</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies the maximum number of records to display in operation, user, service, and rejected reports.</td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td>enrole.ui.report.maxRecordsInReport=5000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Enable/disable WebSEAL single sign-on</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>enrole.ui.ssoEnabled</strong></td>
</tr>
<tr>
<td>The property key and value pair in this section does not pertain to the Tivoli Identity Manager GUI.</td>
</tr>
<tr>
<td>Enable or disables WebSEAL single sign-on functionality. Values are:</td>
</tr>
<tr>
<td>• <strong>true</strong> (enable)</td>
</tr>
<tr>
<td>• <strong>false</strong> (disable)</td>
</tr>
<tr>
<td>Default is &quot;false&quot;.</td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td>enrole.ui.ssoEnabled = false</td>
</tr>
<tr>
<td>Additional configuration is required for WebSEAL single sign-on functionality. See &quot;Configuring Single Sign-on with WebSEAL&quot; on page 40</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Search class mapping for ObjectProfileCategory</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>The property key and value pairs in this section do not pertain to the Tivoli Identity Manager GUI and must not be modified or removed.</td>
</tr>
</tbody>
</table>
### CustomLabels.properties

The property key and value pairs in the CustomLabels.properties file are used by the Tivoli Identity Manager GUI to display the label text for forms.

A separate CustomLabels.properties file exists for each individual language supported by Tivoli Identity Manager.

This file is used to provide localized versions of GUI elements when Tivoli Identity Manager is installed in international environments.

A file name extension identifies the specific language. For example:

- CustomLabels_JA.properties — Japanese
- CustomLabels_EN.properties — English

### Additional property files

The following table lists the remaining property files that are used by Tivoli Identity Manager. In all cases, these files are not configurable and should not be modified.

<table>
<thead>
<tr>
<th>Property File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConfigErrorMessages.properties</td>
<td>This file is used by the runConfig utility and contains configuration error messages in English. This file is not configurable and should not be modified.</td>
</tr>
<tr>
<td>ConfigLabels.properties</td>
<td>This file is used by the runConfig utility and contains Tivoli Identity Manager GUI display labels in English. This file is not configurable and should not be modified.</td>
</tr>
<tr>
<td>ConfigMessages.properties</td>
<td>This file is used by the runConfig utility and contains configuration instructions and normal messages in English. This file is not configurable and should not be modified.</td>
</tr>
<tr>
<td>Dsml2RootDSE.properties</td>
<td>This file is used for performing a root DSE (LDAP) search to return a collection of attributes about the Tivoli Identity Manager server. This file is not configurable and should not be modified.</td>
</tr>
<tr>
<td>Dsml2Schema.properties</td>
<td>This file is used for performing a schema search (LDAP) to return object classes and attributes specified in this file. This file is not configurable and should not be modified.</td>
</tr>
</tbody>
</table>
### enRole2ldif.properties
This file is now deprecated and was used for migration from enRole 3.x to 4.4.

### enRoleEntityHiddenAttributes
This file is used to filter out LDAP attributes for each ITIM entity type available for mapping (for example, Organization, BPOrganization, Person, BPPerson).
Valid ITIM entity types are listed in the Configuration menu of the Tivoli Identity Manager GUI under the Entities tab. Flexible mapping allows the user to map their custom object in LDAP to an ITIM entity type. For example, if you use another Person object rather than inetOrgPerson to store Person information, you must map all ITIM required attributes to the attributes in the custom Person object.

### enRoleFonts.properties
This file is not configurable and should not be modified.

### enRoleHelp.properties
This file is not configurable and should not be modified.

### enRoleHiddenAttributes.properties
This file contains the attributes of each object class (for example, person, service, account, organization unit) that are invisible to the Configuration > User Interface Customization tab of the Tivoli Identity Manager GUI. This hidden attribute list contains mostly attributes used by the system.

### enRoleHiddenSearchAttributes.properties
This file is used in search related functions. Attributes that are in this list do not display in the object class list when performing a search.
Do not remove the existing entries in this file, otherwise the search function on these attributes will fail.

### enRoleLastaccessdate.properties
This file is used by Tivoli Identity Manager internally for dormant report generation.
This file is not configurable and should not be modified.

### enRoleStartup.properties
Modify this file only when configuring a Tivoli Identity Manager functional cluster on WebSphere. Follow the instructions contained in the properties file if you change the Tivoli Identity Manager functional cluster installation.

### enRoleUnchangedAttributes.properties
This file is used by the directory server upgrade utility.
This file is not configurable and should not be modified.

### enRoleValidateAttributes.properties
This file is used internally by the Tivoli Identity Manager server for entity schema attribute mapping.
This file is not configurable and should not be modified.

### ErrorMessages.properties
This file contains all error messages.
This file is not configurable and should not be modified.

### ibmschemaSyntax.properties
This file is used by LDAP configuration during Tivoli Identity Manager installation.
This file is not configurable and should not be modified.

### iplanetSchemaSyntax.properties
This file is used by LDAP configuration during Tivoli Identity Manager installation.
This file is not configurable and should not be modified.

###itiminstaller.properties
This file is used by the Tivoli Identity Manager installer and the directory server upgrade utility.
This file is not configurable and should not be modified.

### Labels.properties
This file contains English labels for the UI display.
This file is not configurable and should not be modified.

### Messages.properties
This file contains all normal messages Tivoli Identity Manager uses to communicate with users.
This file is not configurable and should not be modified.

### passwordrules.properties
This file is used to specify the custom class for generating passwords. Tivoli Identity Manager 4.5.1 has a default password generator and a sample dictionary-based password generator.

In the sample passwordrules.properties, the first line contains the class name. The second line defines the input requires by the class defined in line 1.

### platformcontext.properties
This file is not configurable and should not be modified.

### Properties.properties
This file is the top level properties file that indicates the actual properties file path.

This file is not configurable and should not be modified.

<table>
<thead>
<tr>
<th>tenant.properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>This file used for the creation of a new tenant.</td>
</tr>
<tr>
<td>This file is not configurable and should not be modified.</td>
</tr>
</tbody>
</table>
Part 5. Appendixes
Appendix. 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.
Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged should contact:

IBM Corporation
2ZAA/101
11400 Burnet Road
Austin, TX 78758
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

**Trademarks**

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

AIX
DB2
IBM
IBM logo
SecureWay
Tivoli
Tivoli logo
Universal Database
WebSphere

Lotus is a registered trademark of Lotus Development Corporation and/or IBM Corporation.

Domino is a trademark of International Business Machines Corporation and Lotus Development Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.
Glossary

A

access. The privilege to use information or data stored on computer systems.

account. The set of parameters that define the login information and access control information for a user.

account report. A report that lists people and their associated accounts and whether or not the account is in compliance with current policies.

access control information (ACI). Data that identifies the access rights of a group or principal. See also access control.

ACI origin. The branch in the organization tree where the ACI is created.

ACI target. The set of entities that are controlled by the ACI.

active account. An account that exists and that is in use by the owner to access a resource.

admin domain. A business unit that is used to logically separate organizational responsibilities and manage access rights.

alias. An identity for a user, usually referred to as the user ID. A person can have several aliases, for example: GSmith and GWSmith.

attribute enforcement. The process in which system administrators define the attributes that are required for an account and the values that are valid for those attributes.

audit trail. The record of transactions for a computer system during a given time period.

authentication. The process of identifying an individual, usually based on a user name and password. In security systems, authentication is distinct from authorization, which is the process of giving individuals access to system objects based on their identity. Authentication merely ensures that the individual is who he or she claims to be, but says nothing about the access rights of the individual.

authorization. In computer security, the right granted to a user to communicate with or make use of a computer system. The process of granting a user either complete or restricted access to an object, resource, or function.

Most computer security systems are based on a two-step process. The first stage is authentication, which ensures that a user is who he or she claims to be. The second stage is authorization, which allows the user access to various resources based on the user's identity.

authorization owner. A group of users who can define access control information (ACI) within the context of the organizational unit to which they belong.

B

branch. Each level within the organization tree is called a branch. Each type of branch in the tree is indicated by a different icon. The contents of a branch with sub-units can be viewed by clicking the plus (+) sign next to it.

business partner organization. A class of person that is not a direct employee of the company or organization, but that might need access to the company’s resources.

business partner person. A person in a business partner organization.

business unit. A subsidiary entity of an organization.

C

central data repository. The database used to record and store user and access privilege data for all registered users, including transaction and maintenance records.

Certificate Authority (CA). An organization that issues certificates. The certificate authority authenticates the certificate owner's identity and the services that the owner is authorized to use, issues new certificates, renews existing certificates, and revokes certificates belonging to users who are no longer authorized to use them.

challenge response. An authentication method that requires users to respond to a prompt by providing private information to verify their identity when logging in to the network.

completed requests. Requests that were submitted to the system and that are completed.

constraint. A limitation on a parameter or policy.

control type. An instance of the Java Type class that represents the type of field on a user interface.
**credential.** The User ID and password information for a user, which allows access to an account.

**de-provision.** To remove a service or component. For example, to de-provision an account means to delete an account from a resource.

**digital certificate.** An attachment to an electronic message used for security purposes.

**Directory Services Markup Language (DSML).** An XML implementation that provides a common format for describing and sharing directory services information among different directory systems.

**disallowed action.** A parameter set for reconciliations that defines action to take if the Tivoli Identity Manager Server finds accounts for persons who are not allowed to have an account for the selected service. This parameter is only valid if the Check Policy check box is selected.

**domain administrator.** An administrator that can define and manage provisioning entities, policies, services, workflow definitions, roles, and users within their admin domain, but only in his or her own admin domain.

**DSML identity feed.** One of Tivoli Identity Manager’s three default service types.

A DSML identity feed service imports user data from a human resources database or file and feeds the information into the Tivoli Identity Manager directory. The service can receive the information in one of two ways: a reconciliation or an unsolicited notification.

**escalation participant.** In identity management, a person that has the authority to respond to requests that participants do not respond to within a specified escalation time. An escalation participant can be identified as an individual, as a roles, or by using a custom JavaScript script.

**escalation limit.** The amount of time, in days, hours, minutes or seconds, that a participant has to respond to a request, before an escalation occurs.

**HR feed.** An automated process in which the Tivoli Identity Manager system imports user data from a human resources database or file. Refer to DSML identity feed.

**identity policy.** The rules by which the Tivoli Identity Manager system defines how a user’s ID is created.

**inactive account.** An account that exists in the system, but that is not in use by the account owner.

**ITIM group.** A user group within the Tivoli Identity Manager Server.

System access and administration can be structured around ITIM groups. However, before a person can be assigned to an ITIM group, the user must be provisioned with an ITIM account. Once the person is provisioned with an ITIM account, the person is an ITIM user and can be added to an ITIM group.

**join directive.** The set of rules that define how to handle attributes when two or more provisioning policies conflict.

**keyword.** An index entry that identifies the policy in a search.

**location.** One of the types of subsidiary entities that can be added to an organization. Typically, locations are used to logically separate geographic locations for organizational management purposes.
operation report. A report that lists Tivoli Identity Manager operation requests by type of operation, date, who requested the operation, and who the operation is requested for.

organization. In identity management, a body of users and resources which is fairly independent. Although the sharing of resources between organizations is possible, the level of integration between the organizations is relatively low. Generally, an organization represents a company.

organization tree. A hierarchical structure of the organization that provides a logical place to create, access, and store organizational information.

organizational role. In identity management, an attribute that is used to determine membership to policies that grant access to various managed resources.

organizational unit. A body of users and resources within an organization defined to sub-divide an organization into more manageable groups. Users are assigned to only one organizational unit. Resources are also assigned to only one organizational unit unless they are defined as global to an organization.

orphan (orphan accounts). Accounts on a remote resource whose owner in the Tivoli Identity Manager system cannot be determined.

owner. A person in the Tivoli Identity Manager system that owns an account or a service.

participant. In identity management, a person that has the authority to respond to a request that is submitted through the workflow engine. A participant can be identified as an individual, as a roles, or by using a custom JavaScript script.

password. In computer and network security, a specific string of characters entered by a user and authenticated by the system, which allows the user to gain access to the system and to the information stored within it.

password expiration period. The amount of time a password can be used before the user is forced to change it.

password policy. The rules that define the set parameters that all passwords must meet, such as length, and the type of characters allowed and disallowed.

pending requests. Requests that have been submitted to the system but that have not yet been completed.

personal information. A user’s personal information. This information can include last name, first name, home address, phone number, e-mail address, office number, supervisor, etc.

policy. In Tivoli, a set of rules that are applied to managed resources. For example, a policy can apply to passwords or to resources that a user attempts to access.

policy enforcement. The manner in which the Tivoli Identity Manager system allows or disallows accounts that violate provisioning policies.

provision. To set up and maintain a user’s access to a system in the organization.

provisioning policy. A policy that defines the access to various types of managed services, such as Tivoli Identity Manager or operating systems. Access is granted to all persons or based on a person’s organizational role. Access can also be granted specifically to persons who are not members of any organizational role.

query. A way in which to limit a reconciliation to return smaller packets.

reconciliation. In identity management, the process of synchronizing the accounts and supporting data on the central data repository with the accounts and supporting data on the managed resource.

reconciliation report. A report that lists the orphan accounts found since the last reconciliation was performed.

rejected report. A report that lists requests denied by date, who requested the operation, and who the operation is requested for.

request. An action item in the Tivoli Identity Manager system asking for approval or information.

requestee. The person for whom a request is submitted.

requestor. A person who submits a request.

resource. A hardware, software, or data entity that is managed by Tivoli software. See also managed resource.

resource provisioning management (rpm). The management principle that combines three key elements - business logic, workflow management, and
distribution agents - which together centrally manage the provisioning of users with access to information and business resources.

**restore.** To reactivate an account that was suspended.

**request for information (RFI).** In identity management, an action item that requests additional information from the specified participant and that is a required step in the workflow.

**S**

**scope.** The range that a policy can affect.

Typically, the scope is defined as *single* or *subtree*. When the scope is defined as single, the policy only affects entities in the same branch in which the policy is defined. When the scope is defined as sub-tree, the policy affects the branch in which it is defined and all other branches that are subordinate to the policy’s branch of origin.

**service.** A program that performs a primary function within a server or related software.

**service selection policy.** A JavaScript filter that determines which service to use in a provisioning policy.

**shared secret.** An encrypted value used to retrieve a user’s initial password to access the Tivoli Identity Manager system. This value is defined when the user’s personal information is initially loaded into the system.

**signature authority.** The right to approve or deny a request that is submitted to the workflow engine. A user or group of users is granted signature authority when they are designated as the participant or escalation participant in a workflow design.

**secure socket layer (SSL).** A protocol for transmitting private documents through the Internet. SSL works by using a private key to encrypt data that is transferred over the SSL connection.

**static organizational role.** An organizational role that can only be assigned manually.

**subprocess.** A workflow design that is started as part of another workflow design.

**supervisor.** A person in the Tivoli Identity Manager system that is designated as the owner of a business unit.

**suspend.** The act of deactivating an account so the account owner cannot log into the resource.

**system administrator.** Individuals with access to all areas in the system.

A pre-configured ITIM Group is provided in the Tivoli Identity Manager system. This ITIM Group is designed to grant members maximum access to the system. Users who are members of the administrator ITIM Group have access to all system functions and data.

**T**

**Tivoli Identity Manager Agent.** An intelligent interface between the targeted managed system and the Tivoli Identity Manager Server. It acts as a trusted virtual administrator and is a critical component that translates user requests and provides secure configurations access to various targeted systems.

**Tivoli Identity Manager Server.** A software and services package designed to deploy policy-based provisioning solutions.

**to do list.** The list of actions items assigned to a user for completion.

**U**

**user.** Any person who interacts with the system.

**user class.** An LDAP class such as inetorgperson or BPPerson.

**user interface (UI).** The display used by the user to interact with the system.

**user name.** The ID used by the user to access the system. This ID also identifies the user to the system and allows the system to determine the user’s access rights based on the user’s membership in various organizational roles and ITIM groups.

**user report.** A report that lists all Tivoli Identity Manager operations by date, who requested the operation, and who the operation is requested for.

**W**

**workflow.** The sequence of activities performed in accordance with the business processes of an enterprise.
## Index

**A**
- accessibility statement for documentation  x
- accessing documents online  ix
- adhocreporting.properties file  129
- application client request configuration  125
- application server information  101
- audience, who should read this book  vii
- authentication mechanism, configuring custom  135

**C**
- CA (Certificate Authority)  27
- cache information  105
- Certificate Authority (CA)  27
- Certificate Signing Request (CSR)  27
- certificates (SSL)
  - agent-initiated SSL  36
  - ADK-based agent  36
  - IDI-based agent  36
  - JNDI feed  36
- browser-to-Web-server SSL (WebLogic)  31
- browser-to-Web-server SSL (WebSphere)  27
- Certificate Authority (CA)  27
- Certificate Signing Request (CSR)  27
- configuration  23
- configuration summary  26
- configuring agent certificates  35
- iKeyman utility  27
- key formats  24
- overview  23
- private keys and digital certificates  24
- server-to-agent SSL  33
- CertTool  34, 35
- one-way SSL  34
- SSL implementations  25
- CertTool  34, 35, 36
- changelog  52, 53
- configuring a custom authentication mechanism  135
- contacting software support  x
- conventions used in this document  x
- Crystal Reports  67
- configuring  69
- designing report filter conditions  77
- example reports  79
- process flow  68
- RAS (Report Application Server)  69
- report filter tutorial  77
- crystal.properties file  132
- CSR (Certificate Signing Request)  27
- custom authentication mechanism, configuring  135
- custom display attributes, configuring  16
- custom list displays, configuring  15
- custom logo, configuring  15
- custom reports
  - designing report filter conditions  83
  - example reports  85
  - report filter tutorial  83
- CustomLabels.properties file  158

**D**
- DataBaseFunctions.conf file  133
- default tenant information  103
- display attributes, configuring custom  16
- documents, Tivoli Identity Manager library  vii
  - accessing documents online  ix
  - related documents  ix

**E**
- e-mail message template  19
- encryption information  113
- enRole.properties file  91
-  
  - application client request configuration  125
  - application server information  101
  - cache information  105
  - default tenant information  103
  - encryption information  113
  - LDAP connection pool information  112
  - LDAP server information  104
  - mail services configuration  119
  - messaging information  106
  - password transaction monitor settings  110
  - reconciliation information  120
  - request management UI configuration  124
  - scheduling information  109
  - shared secret hashing  122
  - SSL two-way authentication properties  123
  - system configuration program  114
  - WebLogic-specific configuration  93
  - WebSphere-specific configuration  97
  - workflow configuration information  115
  - XML and DTD information  111
- enRoleAuthentication.properties file  134
- configuring a custom authentication mechanism  135
- enRoleDatabase.properties file  137
- enRoleLDAPConnection.properties file  141
- enRoleLogging.properties file  144
- enRoleMail.properties file  147
- enrolepolicies.properties file  149
- enroletwoflow.properties file  151

**F**
- fesiextensions.properties file  153

**G**
- GSKit (IBM Global Security Toolkit)  25
- GUI
  - custom display attributes  16
  - custom list displays  15
  - custom logo  15

**H**
- heart beat  7
single sign-on (SSO) (continued)
prerequisites, warnings, workarounds 40
URL specification 48
software support, contacting x
SSL
agent-initiated SSL 36
ADK-based agent 36
IDI-based agent 36
JNDI feed 36
browser-to-Web-server SSL (WebLogic) 31
browser-to-Web-server SSL (WebSphere) 27
Certificate Authority (CA) 27
certificate configuration 23
Certificate Signing Request (CSR) 27
configuration summary 26
configuring agent certificates 35
iKeyman utility 27
key formats 24
overview 23
private keys and digital certificates 24
server-to-agent SSL 33
CertTool 34, 35
one-way SSL 34
SSL implementations 25
SSL two-way authentication properties 123
SSO (single sign-on)
configuration procedure 41
configuring with WebSEAL 40
creating a WebSEAL junction 42
SSL junction 45
TCP junction 42
overview 39
plug-in servers configuration 42
prerequisites, warnings, workarounds 40
URL specification 48
supplemental properties 127
adhocreporting.properties file 129
crystal.properties file 132
CustomLabels.properties file 158
DataBaseFunctions.conf file 133
enRoleAuthentication.properties file 134
enRoleDatabase.properties file 137
enRoleLDAPConnection.properties file 141
enRoleLogging.properties file 144
enRoleMail.properties file 147
enrolepolicies.properties file 149
enRoleworkflow.properties file 151
fesiextensions.properties file 153
UI.properties file 155
support, contacting software x
SWebSEAL junction 42
system configuration program 114
system configuration tool
Database tab 8
Database Pool Information 9
General Database Information 9
description 4
Directory tab 7
Directory Server Connection Information 8
LDAP connection pool 8
editing system properties 4
General tab 6
Application Server information 6
Scheduling Information 7
Logging tab 10
Logging Level 10
Trace Level 10
system configuration tool (continued)
Mail tab 10
Host Information 12
Mail Information 12
Web Server Information 11
Security tab 13
application server user management 13
Encryption settings 13
starting 5
UI tab 12
Customer Logo 12
List Page Size 12
system properties
application client request configuration 125
application server information 101
cache information 105
default tenant information 103
encryption information 113
enRole.properties file 91
LDAP connection pool information 112
LDAP server information 104
mail services configuration 119
messaging information 106
password transaction monitor settings 110
reconciliation information 120
request management UI configuration 124
scheduling information 109
shared secret hashing 122
SSL two-way authentication properties 123
system configuration program 114
understanding properties files 92, 128
WebLogic-specific configuration 93
WebSphere-specific configuration 97
workflow configuration information 115
XML and DTD information 111

T
Tivoli Identity Manager GUI
configuring properties 14

U
UI.properties file 155

W
WebLogic-specific configuration 93
WebSEAL, configuring SSO 40
WebSphere-specific configuration 97
workflow configuration information 115

X
XML and DTD information 111
Program Number: 5724–C34

Printed in USA