Problem Determination Guide

Version 5.1
Problem Determination Guide

Version 5.1
Before using this information and the product it supports, read the information in Appendix E, “Notices,” on page 127.
## Contents

### Preface
- Who should read this book .................................................. vii
- What this book contains ...................................................... vii

### Publications
- Release information .......................................................... viii
- Base information ............................................................. viii
- Web security information .................................................... ix
- Developer references ......................................................... ix
- Technical supplements ....................................................... x
- Related publications .......................................................... x
- Accessing publications online ............................................... xiii

### Accessibility ................................................................. xiii
Contacting software support .................................................. xiv
Conventions used in this book ................................................ xiv
- Typeface conventions .......................................................... xiv
- Operating system differences ............................................... xiv

### Chapter 1. Introduction to problem determination .......... 1
- Problem determination overview ......................................... 1
- Avoiding potential problems .............................................. 1
- Problem resolution ............................................................ 2
- Product identifiers ............................................................ 2
- Serviceability improvements .............................................. 3
  - Tivoli Common Directory .................................................. 3
  - Tivoli Message Standard .................................................. 3
  - Tivoli Log XML format .................................................... 3
- Application specific logging for Java applications ................. 3

### Chapter 2. Gathering initial diagnostic information .... 5
- Locating tools and utilities .................................................. 5
- Obtaining version information ............................................ 5
  - Product version information (pdversion) ............................... 5
  - IBM Global Security Kit (GSKit) version information ............. 6
  - IBM Tivoli Directory Client version information ................... 6
  - LDAP server version information ...................................... 6
- Gathering system information (pdbackup) ............................... 8
  - Gathering base information ............................................. 8
  - Gathering WebSEAL information ....................................... 8
  - Gathering Plug-in for Web Servers information ..................... 9
- Validating and maintaining policy databases (pdacld_dump) ....... 9
  - Displaying the entire database ....................................... 10
  - Summary of a policy or replica database ............................. 12
  - Repairing a damaged policy database ................................ 13

### Chapter 3. Troubleshooting installation problems ........ 15
- Problems with installation ................................................ 15
  - Problem with insufficient space for temporary files .......... 15
  - Problem connecting to display ....................................... 15
  - Removing the ibmjcaprovider.jar file ................................. 15
  - Problems running the installation program ......................... 16
  - Problems installing and configuring the product .................. 16
- Native installation log files ............................................. 17
- Native configuration log files .......................................... 17

### Chapter 4. Default message logging ......................... 19
Chapter 8. Using WebSEAL statistics ........................................ 75
   pdadmin stats command syntax ..................................... 75
      Basic pdadmin stats command ................................ 75
      Enable statistics dynamically ................................ 75
      Disable statistics .............................................. 77
      Show enabled statistics components ......................... 77
      Get current statistic values dynamically .................... 77
      Reset statistics values ..................................... 77
      List all available statistics components .................... 78
   Statistics components and activity types ....................... 78
      pdweb.authn component ....................................... 78
      pdweb.authz component ..................................... 78
      pdweb.http component ....................................... 79
      pdweb.https component ...................................... 79
      pdweb.threads component ................................... 80
      pdweb.jnt component ....................................... 80
      pdweb.sescache component .................................. 80
      pdweb.docache component ................................... 81
      pdweb.jct.# component ..................................... 83
   Enabling statistics statically using event logging .......... 83

Chapter 9. Basic troubleshooting ..................................... 85
   Verifying software installation .................................. 85
   Verifying the functionality of individual components .......... 86
      IBM Global Security Kit (GSKit) ............................ 87
      LDAP server ................................................. 87
      IBM Tivoli Directory Client .................................. 88
      Microsoft Active Directory .................................. 89
      Lotus Domino Server ........................................ 90
      Tivoli Access Manager servers ................................ 90
      Tivoli Access Manager policy server ......................... 91
      Tivoli Access Manager authorization server ................ 91
      Tivoli Access Manager runtime ................................ 91
      Tivoli Access Manager WebSEAL ............................... 91
      Junctioned third-party Web server ............................ 92
      Tivoli Access Manager Plug-in for Web Servers ............... 92

Chapter 10. Solutions to common problems. ......................... 95
   Tivoli Access Manager Base common problems .................. 95
      Unable to configure the policy server ....................... 95
      Unable to create new user .................................... 97
      Invalid account ............................................. 97
      Insufficient disk space causes problems ..................... 98
      Unexpected permit/deny access to resources ................. 98
   LDAP common problems ......................................... 100
      LDAP does not start after secAuthority=Default suffix is created ........................................ 100
      Insufficient LDAP access privileges to perform operation ........................................ 101
      IBM Tivoli Directory Server error log warnings ............. 101
   Tivoli Access Manager WebSEAL common problems ............... 101
      WebSEAL does not respond on ports 80 or 443 ......... 102
      Multiple logins required during e-community ............... 102
Active Directory password complexity rules enforced even when disabled .......................... 102
Web Security common problems .................................................................................. 102
Upgrade common problems ....................................................................................... 103
Cannot create users and groups after upgrade ............................................................ 103

**Chapter 11. Contacting customer support** ................................................................. 105
General data to collect for customer support ............................................................... 105
Collecting defect information for pdadmin and C APIs ............................................. 105
  - Basic procedure ..................................................................................................... 105
  - Getting the pdmgrd trace file ............................................................................. 106
  - Getting the pdacld trace file ............................................................................. 106
  - Getting the pdmgrproxyd trace file ................................................................. 106
  - Getting the pdadmin and C API trace file ....................................................... 106
  - Getting the message files ................................................................................. 106
Collecting defect information for Java APIs .............................................................. 107
  - Basic procedure .................................................................................................. 107
  - Enabling tracing and messaging for Java runtime environment components ..... 107
  - Getting the pdmgrd trace file ............................................................................. 107
  - Getting the pdacld trace file ............................................................................. 107
  - Getting the pdmgrproxyd trace file ................................................................. 107
  - Getting the Java API trace files ....................................................................... 108

**Appendix A. Example routing file** ................................................................. 109

**Appendix B. Sample Java application properties file** ......................................... 111

**Appendix C. Sample PDJLog.properties file** ................................................... 115

**Appendix D. User registry differences** ............................................................. 123

**Appendix E. Notices** ......................................................................................... 127
  Trademarks ............................................................................................................. 129

**Index** .................................................................................................................. 131
Preface

Welcome to the IBM® Tivoli® Access Manager Problem Determination Guide.

IBM® Tivoli® Access Manager (Tivoli Access Manager) is the base software that is required to run applications in the IBM Tivoli Access Manager product suite. It enables the integration of IBM Tivoli Access Manager applications that provide a wide range of authorization and management solutions. Sold as an integrated solution, these products provide an access control management solution that centralizes network and application security policy for e-business applications.

Note: IBM Tivoli Access Manager is the new name of the previously released software entitled Tivoli SecureWay® Policy Director. Also, for users familiar with the Tivoli SecureWay Policy Director software and documentation, the management server is now referred to as the policy server.

This problem determination guide provides a comprehensive set of procedures and reference information for troubleshooting Tivoli Access Manager.

Who should read this book

This guide is for system administrators and field support personnel responsible for troubleshooting a Tivoli Access Manager environment.

Readers should be familiar with the following:
- PC and UNIX® operating systems
- Database architecture and concepts
- Security management
- Internet protocols, including HTTP, TCP/IP, File Transfer Protocol (FTP), and Telnet
- Lightweight Directory Access Protocol (LDAP) and directory services
- A supported user registry
- Authentication and authorization
- Secure Sockets Layer (SSL) protocol, key exchange (public and private), digital signatures, cryptographic algorithms, and certificate authorities.

What this book contains

This book contains the following chapters:
- Chapter 1, “Introduction to problem determination,” on page 1
- Chapter 2, “Gathering initial diagnostic information,” on page 5
- Chapter 3, “Troubleshooting installation problems,” on page 15
- Chapter 4, “Default message logging,” on page 19
- Chapter 5, “Tailoring message logging,” on page 27
- Chapter 6, “Trace logging,” on page 43
- Chapter 7, “Using AutoTrace,” on page 59
- Chapter 8, “Using WebSEAL statistics,” on page 75
- Chapter 9, “Basic troubleshooting,” on page 85
Publications

Review the descriptions of the Tivoli Access 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.

Additional information about the IBM Tivoli Access Manager for e-business product itself can be found at:


The Tivoli Access Manager library is organized into the following categories:

- "Release information"
- "Base information"
- "Web security information" on page ix
- "Developer references" on page ix
- "Technical supplements" on page x

Release information

- **IBM Tivoli Access Manager for e-business Read This First (GI11-4155-00)**
  Provides information for installing and getting started using Tivoli Access Manager.

- **IBM Tivoli Access Manager for e-business Release Notes (GI11-4156-00)**
  Provides late-breaking information, such as software limitations, workarounds, and documentation updates.

Base information

- **IBM Tivoli Access Manager Base Installation Guide (SC32-1362-00)**
  Explains how to install and configure the Tivoli Access Manager base software, including the Web Portal Manager interface. This book is a subset of **IBM Tivoli Access Manager for e-business Web Security Installation Guide** and is intended for use with other Tivoli Access Manager products, such as IBM Tivoli Access Manager for Business Integration and IBM Tivoli Access Manager for Operating Systems.

- **IBM Tivoli Access Manager Base Administration Guide (SC32-1360-00)**
  Describes the concepts and procedures for using Tivoli Access Manager services. Provides instructions for performing tasks from the Web Portal Manager interface and by using the `pdadmin` command.
Web security information

- **IBM Tivoli Access Manager for e-business Web Security Installation Guide** (SC32-1361-00)
  Provides installation, configuration, and removal instructions for the Tivoli Access Manager base software as well as the Web Security components. This book is a superset of *IBM Tivoli Access Manager Base Installation Guide*.

- **IBM Tivoli Access Manager Upgrade Guide** (SC32-1369-00)
  Explains how to upgrade from Tivoli SecureWay Policy Director Version 3.8 or previous versions of Tivoli Access Manager to Tivoli Access Manager Version 5.1.

- **IBM Tivoli Access Manager for e-business WebSEAL Administration Guide** (SC32-1359-00)
  Provides background material, administrative procedures, and technical reference information for using WebSEAL to manage the resources of your secure Web domain.

- **IBM Tivoli Access Manager for e-business WebSphere Application Server Integration Guide** (SC32-1368-00)
  Provides installation, removal, and administration instructions for integrating Tivoli Access Manager with IBM WebSphere® Application Server.

- **IBM Tivoli Access Manager for e-business IBM WebSphere Edge Server Integration Guide** (SC32-1367-00)
  Provides installation, removal, and administration instructions for integrating Tivoli Access Manager with the IBM WebSphere Edge Server application.

- **IBM Tivoli Access Manager for e-business Plug-in for Web Servers Integration Guide** (SC32-1365-00)
  Provides installation instructions, administration procedures, and technical reference information for securing your Web domain using the plug-in for Web servers.

- **IBM Tivoli Access Manager for e-business BEA WebLogic Server Integration Guide** (SC32-1366-00)
  Provides installation, removal, and administration instructions for integrating Tivoli Access Manager with BEA WebLogic Server.

- **IBM Tivoli Access Manager for e-business IBM Tivoli Identity Manager Provisioning Fast Start Guide** (SC32-1364-00)
  Provides an overview of the tasks related to integrating Tivoli Access Manager and Tivoli Identity Manager and explains how to use and install the Provisioning Fast Start collection.

Developer references

- **IBM Tivoli Access Manager for e-business Authorization C API Developer Reference** (SC32-1355-00)
  Provides reference material that describes how to use the Tivoli Access Manager authorization C API and the Tivoli Access Manager service plug-in interface to add Tivoli Access Manager security to applications.

- **IBM Tivoli Access Manager for e-business Authorization Java Classes Developer Reference** (SC32-1350-00)
  Provides reference information for using the Java™ language implementation of the authorization API to enable an application to use Tivoli Access Manager security.
• IBM Tivoli Access Manager for e-business Administration C API Developer Reference (SC32-1357-00)
Provides reference information about using the administration API to enable an application to perform Tivoli Access Manager administration tasks. This document describes the C implementation of the administration API.

• IBM Tivoli Access Manager for e-business Administration Java Classes Developer Reference (SC32-1356-00)
Provides reference information for using the Java language implementation of the administration API to enable an application to perform Tivoli Access Manager administration tasks.

• IBM Tivoli Access Manager for e-business Web Security Developer Reference (SC32-1358-00)
Provides administration and programming information for the cross-domain authentication service (CDAS), the cross-domain mapping framework (CDMF), and the password strength module.

Technical supplements

• IBM Tivoli Access Manager for e-business Command Reference (SC32-1354-00)
Provides information about the command line utilities and scripts provided with Tivoli Access Manager.

• IBM Tivoli Access Manager Error Message Reference (SC32-1353-00)
Provides explanations and recommended actions for the messages produced by Tivoli Access Manager.

• IBM Tivoli Access Manager for e-business Problem Determination Guide (SC32-1352-00)
Provides problem determination information for Tivoli Access Manager.

• IBM Tivoli Access Manager for e-business Performance Tuning Guide (SC32-1351-00)
Provides performance tuning information for an environment consisting of Tivoli Access Manager with the IBM Tivoli Directory server as the user registry.

Related publications
This section lists publications related to the Tivoli Access Manager library.

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

IBM Global Security Kit
Tivoli Access Manager provides data encryption through the use of the IBM Global Security Kit (GSKit) Version 7.0. GSKit is included on the IBM Tivoli Access Manager Base CD for your particular platform, as well as on the IBM Tivoli Access Manager Web Security CDs, the IBM Tivoli Access Manager Web Administration Interfaces CDs, and the IBM Tivoli Access Manager Directory Server CDs.

The GSKit package provides the iKeyman key management utility, gsk7iKm, which is used to create key databases, public-private key pairs, and certificate requests.
The following document is available on the Tivoli Information Center Web site in the same section as the IBM Tivoli Access Manager product documentation:

- **IBM Global Security Kit Secure Sockets Layer and iKeyman User’s Guide**
  (SC32-1363-00)
  Provides information for network or system security administrators who plan to enable SSL communication in their Tivoli Access Manager environment.

**IBM Tivoli Directory Server**
IBM Tivoli Directory Server, Version 5.2, is included on the **IBM Tivoli Access Manager Directory Server** CD for the desired operating system.

**Note:** IBM Tivoli Directory Server is the new name for the previously released software known as:
  - IBM Directory Server (Version 4.1 and Version 5.1)
  - IBM SecureWay Directory Server (Version 3.2.2)


Additional information about IBM Tivoli Directory Server can be found at:


**IBM DB2 Universal Database**
IBM DB2® Universal Database™ Enterprise Server Edition, Version 8.1 is provided on the **IBM Tivoli Access Manager Directory Server** CD and is installed with the IBM Tivoli Directory Server software. DB2 is required when using IBM Tivoli Directory Server, z/OS™, or OS/390® LDAP servers as the user registry for Tivoli Access Manager.

Additional information about DB2 can be found at:


**IBM WebSphere Application Server**
IBM WebSphere Application Server, Advanced Single Server Edition 5.0, is included on the **IBM Tivoli Access Manager Web Administration Interfaces** CD for the desired operating system. WebSphere Application Server enables the support of both the Web Portal Manager interface, which is used to administer Tivoli Access Manager, and the Web Administration Tool, which is used to administer IBM Tivoli Directory Server. IBM WebSphere Application Server Fix Pack 2 is also required by Tivoli Access Manager and is provided on the **IBM Tivoli Access Manager WebSphere Fix Pack** CD.

Additional information about IBM WebSphere Application Server can be found at:


**IBM Tivoli Access Manager for Business Integration**
IBM Tivoli Access Manager for Business Integration, available as a separately orderable product, provides a security solution for IBM MQSeries®, Version 5.2, and IBM WebSphere® MQ for Version 5.3 messages. IBM Tivoli Access Manager for Business Integration allows WebSphere MQSeries applications to send data with privacy and integrity by using keys associated with sending and receiving applications. Like WebSEAL and IBM Tivoli Access Manager for Operating Systems, Tivoli Access Manager for Business Integration enables system administrators to secure and manage access to resources at the point of user authentication.
Systems, IBM Tivoli Access Manager for Business Integration, is one of the resource managers that use the services of IBM Tivoli Access Manager.

Additional information about IBM Tivoli Access Manager for Business Integration can be found at:


The following documents associated with IBM Tivoli Access Manager for Business Integration Version 5.1 are available on the Tivoli Information Center Web site:

- IBM Tivoli Access Manager for Business Integration Administration Guide (SC23-4831-01)
- IBM Tivoli Access Manager for Business Integration Problem Determination Guide (GC23-1328-00)
- IBM Tivoli Access Manager for Business Integration Release Notes (GI11-0957-01)
- IBM Tivoli Access Manager for Business Integration Read This First (GI11-4202-00)

IBM Tivoli Access Manager for WebSphere Business Integration Brokers
IBM Tivoli Access Manager for WebSphere Business Integration Brokers, available as part of IBM Tivoli Access Manager for Business Integration, provides a security solution for WebSphere Business Integration Message Broker, Version 5.0 and WebSphere Business Integration Event Broker, Version 5.0. IBM Tivoli Access Manager for WebSphere Business Integration Brokers operates in conjunction with Tivoli Access Manager to secure JMS publish/subscribe applications by providing password and credentials-based authentication, centrally-defined authorization, and auditing services.

Additional information about IBM Tivoli Access Manager for WebSphere Integration Brokers can be found at:


The following documents associated with IBM Tivoli Access Manager for WebSphere Integration Brokers, Version 5.1 are available on the Tivoli Information Center Web site:

- IBM Tivoli Access Manager for WebSphere Business Integration Brokers Administration Guide (SC32-1347-00)
- IBM Tivoli Access Manager for WebSphere Business Integration Brokers Release Notes (GI11-4154-00)
- IBM Tivoli Access Manager for Business Integration Read This First (GI11-4202-00)

IBM Tivoli Access Manager for Operating Systems
IBM Tivoli Access Manager for Operating Systems, available as a separately orderable product, provides a layer of authorization policy enforcement on UNIX systems in addition to that provided by the native operating system. IBM Tivoli Access Manager for Operating Systems, like WebSEAL and IBM Tivoli Access Manager for Business Integration, is one of the resource managers that use the services of IBM Tivoli Access Manager.

Additional information about IBM Tivoli Access Manager for Operating Systems can be found at:

The following documents associated with IBM Tivoli Access Manager for Operating Systems Version 5.1 are available on the Tivoli Information Center Web site:

- **IBM Tivoli Access Manager for Operating Systems Installation Guide** (SC23-4829-00)
- **IBM Tivoli Access Manager for Operating Systems Administration Guide** (SC23-4827-00)
- **IBM Tivoli Access Manager for Operating Systems Problem Determination Guide** (SC23-4828-00)
- **IBM Tivoli Access Manager for Operating Systems Release Notes** (GI11-0951-00)
- **IBM Tivoli Access Manager for Operating Systems Read Me First** (GI11-0949-00)

**IBM Tivoli Identity Manager**

IBM Tivoli Identity Manager Version 4.5, available as a separately orderable product, enables you to centrally manage users (such as user IDs and passwords) and provisioning (that is providing or revoking access to applications, resources, or operating systems.) Tivoli Identity Manager can be integrated with Tivoli Access Manager through the use of the Tivoli Access Manager Agent. Contact your IBM account representative for more information about purchasing the Agent.

Additional information about IBM Tivoli Identity Manager can be found 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: [http://www.ibm.com/software/tivoli/library](http://www.ibm.com/software/tivoli/library)

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

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**

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

**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/](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.
Chapter 1. Introduction to problem determination

Problem determination, or troubleshooting, is a process of determining why a product is not functioning in the expected manner. This guide provides information to aid in the identification and resolution of problems encountered when using IBM Tivoli Access Manager (Tivoli Access Manager) and its associated prerequisite products.

Problem determination overview

Tivoli Access Manager provides an authentication and authorization framework for permitting or restricting access to system resources located in a secure domain. This guide provides information about Tivoli Access Manager tools, resources, and techniques that can aid in the identification and resolution of problems.

Subjects discussed include:
• Diagnostic and data collection tools
• Message logs
• Tracing tools and logs
• Product statistics
• Basic troubleshooting information
• Solutions to common Tivoli Access Manager problems

Avoiding potential problems

Problems can often be avoided with planning and preparation before deploying the Tivoli Access Manager software. Before installing Tivoli Access Manager, review the IBM Tivoli Access Manager for e-business Release Notes and the IBM Tivoli Access Manager for e-business Web Security Installation Guide. These documents contain the following important information:
• Supported operating system levels
• Prerequisite software requirements
• Required software patches
• Minimum and recommended memory requirements
• Disk space requirements
• Upgrade considerations
• Known problems, limitations, and recovery procedures
• Customer support contact information

After you have installed Tivoli Access Manager, ensure that you have a comprehensive backup and system recovery strategy in place. Further avoid the possibility of running into problems by doing the following:
• Perform regular periodic backups of Tivoli Access Manager using the pdbackup command.
• Periodically backup the user registry following the instructions provided by the user registry vendor.
• Maintain information on your environment, including system topology, IP addresses, host names, and what components are installed on each system.
• Maintain updated information describing the key system resources being managed by Tivoli Access Manager and the security policies being applied to them by Tivoli Access Manager.

• Periodically check that all systems running Tivoli Access Manager have sufficient disk space for runtime and problem determination data. As your security policy grows, and the number of users, groups, and protected objects increase, the space requirements for the policy databases, message logs, trace logs, and any auditing information can increase as well.

• Regularly check for the availability of fix packs and install them as they become available. Information on fix packs and other usual information can be found on the IBM Tivoli Software Support site by clicking the Tivoli support link at the following Web site:

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

Problem resolution

When problems do occur, use the information in this guide to identify and possibly resolve them. If you are unable to correct the problem, gather the relevant diagnostic information as described in this guide and then use the information in Chapter 11, “Contacting customer support,” on page 105 to get further assistance.

Product identifiers

To provide consistency with other IBM and Tivoli products, components of Tivoli Access Manager identify some product-specific information with a unique product identifier. This three character identifier is used in the following places:

• As the first three characters of message IDs. (See the IBM Tivoli Access Manager Error Message Reference for a complete explanation of messages in Tivoli Access Manager)

• To identify the subdirectory containing serviceability information when using the Tivoli Common Directory

• AutoTrace (see Chapter 7, “Using AutoTrace,” on page 59 for details)

The product identifiers associated with Tivoli Access Manager are shown in Table 1.

<table>
<thead>
<tr>
<th>Product identifier</th>
<th>Tivoli Access Manager Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPD</td>
<td>Base and Web Portal Manager</td>
</tr>
<tr>
<td>DPW</td>
<td>WebSEAL (see Note)</td>
</tr>
<tr>
<td>AWD</td>
<td>Plug-in for IBM WebSphere Edge Server</td>
</tr>
<tr>
<td>AWL</td>
<td>BEA WebLogic Server integration</td>
</tr>
<tr>
<td>AWX</td>
<td>WebSphere Application Server integration</td>
</tr>
<tr>
<td>AMZ</td>
<td>Plug-in for Web Servers</td>
</tr>
</tbody>
</table>

Note: WebSEAL uses an identifier of HPW to identify its subdirectory when using the Tivoli Common Directory.
Serviceability improvements

Tivoli is committed to improving the serviceability of its products. In this release of Tivoli Access Manager, the following Tivoli serviceability initiatives have been implemented:

- Tivoli Common Directory
- Tivoli Message Standard
- Tivoli Log XML format

Tivoli Common Directory

To provide a consistent mechanism for locating serviceability information, some Tivoli products and applications have implemented the Tivoli Common Directory. The Tivoli Common Directory represents a central location on systems running Tivoli software for storing serviceability-related files, such as trace logs, message logs, first failure data capture information, as well as serviceability scripts.

Tivoli Access Manager does not take advantage of the Tivoli Common Directory unless you explicitly request this behavior during the installation of the product. By default, serviceability information is stored in the log subdirectory of the product installation directory.

If Tivoli Common Directory support is requested, the installation program uses the existing Tivoli Common Directory as the default location for serviceability information. If no existing Tivoli Common Directory is in use, the directory specified during the installation is identified as the Tivoli Common Directory and serviceability information for Tivoli Access Manager and other Tivoli products is stored there.

Tivoli Message Standard

All messages issued by Tivoli Access Manager adhere to the Tivoli Message Standard. The Tivoli Message Standard specifies a standard format for all messages issued by Tivoli products. The standard, based on the IBM Message Standard, is intended to provide a consistent and meaningful way for identifying messages across the entire Tivoli product set. Messages issued by Tivoli Access Manager, along with detailed explanations and suggested actions, can be found in the IBM Tivoli Access Manager Error Message Reference.

Tivoli Log XML format

The C language based components of Tivoli Access Manager support the generation of message and trace information in a common XML format. This common format is known as the Tivoli Log XML format and is used by a number of Tivoli applications. A Java language-based log viewer is provided that allows these messages and traces to be filtered in a number of ways, including by time window, severity, thread ID, and component. Information produced by different products can be analyzed and converted into ASCII or HTML using the Tivoli Log XML viewer.

The message and trace logs produced by the Java language based components of Tivoli Access Manager do not support the generation of Tivoli Log XML data.

Application specific logging for Java applications

Configuration of message and trace logging for the components of the Access Manager Java Runtime Environment (AMJRTE) is now performed on a
per-application basis. This removes the file contention and ownership problems encountered in previous versions of Tivoli Access Manager. In addition to existing configuration properties, the application properties file created by the `com.tivoli.pd.jcfg.SvrSslCfg` command contains logging properties associated with the application. The names of the logging objects and the log files in this configuration file contain the application server name supplied by the `appSvr` parameter of `com.tivoli.pd.jcfg.SvrSslCfg`. Thus, each application has a unique set of objects and log files. If an application configuration file is not being used (for instance, when the `pdjrtecfg` command is used), message logging and tracing properties are taken from the existing PDJLog.properties file.

In addition, the size and number of files used for messages and trace entries are now configurable.
Chapter 2. Gathering initial diagnostic information

One of the first steps in diagnosing a problem with IBM Tivoli Access Manager (Tivoli Access Manager) is to determine the state of your security environment. This includes locating the diagnostic tools and utilities, as well as determining what products (and what versions of those products) are installed and configured.

Problems installing and initially configuring the product are addressed in Chapter 3, “Troubleshooting installation problems,” on page 15.

Locating tools and utilities

The installation directory for Tivoli Access Manager Base is specified when the product is installed on a system. The default location for the product files and directories is platform dependent:

**Microsoft Windows**  
C:\Program Files\Tivoli\Policy Director

**UNIX**  
/opt/PolicyDirector

**Note:** As indicated in “Operating system differences” on page xiv, the UNIX conventions are used for directories and environment variables in this guide where there is no difference between operating system platforms. Items which are different between different operating system platforms, such as the location of the installation directory on UNIX and Microsoft Windows systems, are indicated as shown.

The environment variable PD_HOME is set to the installation directory on Windows systems. No environment variable is set on UNIX systems. Ensure that only trusted users and groups have access to the installation directory and its subdirectories.

Many of the commands, tools, scripts, and daemons associated with Tivoli Access Manager Base are installed under the installation directory in the bin and sbin subdirectories. In order to run most of the commands shown in this guide, you must have access to these directories and their files. Commands can be run from any command prompt or shell. You can explicitly change to the desired directory and run the command, or you can add the two directories to your command search path, which enables the commands to be run from any directory.

The Tivoli Log XML Viewer is one of the exceptions. This viewer is installed separately and resides in its own directory if the default installation location is accepted. See “Tivoli XML Log Viewer” on page 39 for details.

Obtaining version information

Use the information in the following sections to determine the version of the various components and products installed in a Tivoli Access Manager environment.

**Product version information (pdversion)**

The pdversion command, located under the installation directory in the bin subdirectory, displays a list of IBM Tivoli Access Manager components and indicates the version number for any components installed on the system.
Sample output from the command is as follows:

<table>
<thead>
<tr>
<th>Product</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Tivoli Access Manager Runtime</td>
<td>5.1.0.0</td>
</tr>
<tr>
<td>IBM Tivoli Access Manager Policy Server</td>
<td>5.1.0.0</td>
</tr>
<tr>
<td>IBM Tivoli Access Manager Web Portal Manager</td>
<td>5.1.0.0</td>
</tr>
<tr>
<td>IBM Tivoli Access Manager Application Developer Kit</td>
<td>5.1.0.0</td>
</tr>
<tr>
<td>IBM Tivoli Access Manager Authorization Server</td>
<td>Not Installed</td>
</tr>
<tr>
<td>IBM Tivoli Access Manager Java Runtime Environment</td>
<td>5.1.0.0</td>
</tr>
<tr>
<td>IBM Tivoli Access Manager Policy Proxy Server</td>
<td>Not Installed</td>
</tr>
</tbody>
</table>

### IBM Global Security Kit (GSKit) version information

Secure Sockets Layer (SSL) communication in Tivoli Access Manager is provided by the IBM Global Security Kit (GSKit). Each version of Tivoli Access Manager potentially provides a different level of GSKit. In addition, updates to GSKit might be applied as a result of applying fix packs or other service. In some cases, other versions of GSKit might be installed with other IBM products.

To determine the version of GSKit installed, use the `gsk7ver` command, located within the GSKit installation directory tree. The default location for each operating system platform is as follows:

- **AIX**  `/usr/bin`
- **HP-UX** `/usr/bin`
- **Linux** `/usr/local/ibm/gsk7/bin`
- **Solaris** `/usr/bin`
- **Windows** `C:\Program Files\IBM\gsk7\bin`

The `gsk7ver` command invokes all of the GSKit shared libraries and displays version information about each library.

**Note:** Previous versions of Tivoli Access Manager provided a different level of GSKit which provided a similar command, called `gsk5ver`. The `gsk7ver` command must now be used.

### IBM Tivoli Directory Client version information

The IBM Tivoli Directory Client is used by Tivoli Access Manager to communicate with any LDAP-based user registry, not just with the IBM Tivoli Directory Server. The client is not needed if Microsoft Active Directory or Lotus Domino server is being used as the Tivoli Access Manager user registry. The IBM Tivoli Directory Client is installed on any system that communicates with an LDAP-based user registry.

To determine the version of the IBM Tivoli Directory Client installed, use the `ldapsearch` command:

```
ldapsearch -e
```

### LDAP server version information

If an LDAP-based user registry, such as IBM Tivoli Directory Server, Netscape iPlanet Directory Server, or Novell eDirectory, is being used as the Tivoli Access Manager user registry, you can use the following `ldapsearch` command from any system with the IBM Tivoli Directory Client installed to determine the version of the server:

```
ldapsearch -h ldapserver_host_name -p 389 -D "cn=root" -w password -b "**" -s base objectclass=""
```
ldapsrvr_host_name

The host name for the LDAP user registry server.

389

The port number used for communications with the server. This is the default value.

cn=root

The distinguished name (DN) for the administrator. This is the default value.

password

The password of the administrator.

The output of this command when IBM Tivoli Directory Server Version 5.2 is being used as the user registry would be similar to the following:

namingcontexts=CN=SCHEMA
namingcontexts=CN=LOCALHOST
namingcontexts=CN=PWDPOLICY
namingcontexts=CN=IBMPOLICIES
namingcontexts=O=IBM,C=US
namingcontexts=SECAUTHORITY=DEFAULT
namingcontexts=CN=CONFIGURATION
ibm-configurationnamingcontext=CN=CONFIGURATION
subschemasubentry=cn=schema
supportedextension=1.3.18.0.2.12.1
supportedextension=1.3.18.0.2.12.3
supportedextension=1.3.18.0.2.12.5
supportedextension=1.3.18.0.2.12.6
supportedextension=1.3.18.0.2.12.15
supportedextension=1.3.18.0.2.12.16
supportedextension=1.3.18.0.2.12.17
supportedextension=1.3.18.0.2.12.19
supportedextension=1.3.18.0.2.12.44
supportedextension=1.3.18.0.2.12.24
supportedextension=1.3.18.0.2.12.22
supportedextension=1.3.18.0.2.12.20
supportedextension=1.3.18.0.2.12.28
supportedextension=1.3.18.0.2.12.30
supportedextension=1.3.18.0.2.12.26
supportedextension=1.3.6.1.4.1.1466.20037
supportedextension=1.3.18.0.2.12.35
supportedextension=1.3.18.0.2.12.40
supportedextension=1.3.18.0.2.12.46
supportedextension=1.3.18.0.2.12.37
supportedcontrol=2.16.840.1.113730.3.4.2
supportedcontrol=1.3.18.0.2.10.5
supportedcontrol=1.2.840.113556.1.4.473
supportedcontrol=1.2.840.113556.1.4.319
supportedcontrol=1.3.6.1.4.1.42.2.27.8.5.1
supportedcontrol=1.2.840.113556.1.4.805
supportedcontrol=2.16.840.1.113730.3.4.18
supportedcontrol=1.3.18.0.2.10.15
supportedcontrol=1.3.18.0.2.10.18
security=none
port=389
supportedasmechanisms=CRAM-MD5
supportedasmechanisms=DIGEST-MD5
supportedldapversion=2
supportedldapversion=3
ibmdirectoryversion=5.2
ibm-ldapservicename=ldapsrvr.somewhere.com
ibm-serverId=fa8deae4-4aee-4a4e-4b6c-bb2c-4b5fe138927
ibm-supportedacimechanisms=1.3.18.0.2.26.3
ibm-supportedacimechanisms=1.3.18.0.2.26.4
ibm-supportedacimechanisms=1.3.18.0.2.26.2
vendorname=International Business Machines (IBM)
vendorversion=5.2
ibm-sslciphers=N/A

Chapter 2. Gathering initial diagnostic information  7
ibm-slapdisconfigurationmode=FALSE
ibm-slapdSizeLimit=500
ibm-slapdTimeLimit=900
ibm-slapdDerefAliases=always
ibm-supportedAuditVersion=2
ibm-sasldigestrealmname=ldapserver.somewhere.com

For the IBM Tivoli Directory Server, the version information also can be found in
the following file:
IDS_install_directory/web/buildno.txt

Gathering system information (pdbackup)

This section describes how to use the pdbackup command, located under the
installation directory in the bin subdirectory, to gather diagnostic information
about your Tivoli Access Manager system. Comprehensive syntax and reference
information on the pdbackup command can be found in the IBM Tivoli Access
Manager for e-business Command Reference. This section only describes the use of the
pdbackup command with the Tivoli Access Manager supplied service list files for
gathering system information.

Note: In previous versions of Tivoli Access Manager, the pdinfo command was
used for this purpose. The pdinfo command is no longer provided as part
of the product.

Gathering base information

To gather information about Tivoli Access Manager Base, use the
install_dir/etc/pdinfo.1st service list file with the pdbackup command. For
example:
pbackup -action backup -list /opt/PolicyDirector/etc/pdinfo.1st

On UNIX systems, this command invocation creates a file called
/var/PolicyDirector/pdbackup/pdinfo.1st_ddmmyyyy.hh_mm.tar, where the
current date and time are substituted for ddmmyyyy and hh_mm, respectively. A
representative file name would be pdinfo.1st_200ct2003.23_20.tar.

Similarly, on Windows systems, the file would be called
%PD_HOME%\pdbackup\pdinfo.1st_ddmmyyyy.hh_mm.dar.

Messages associated with the running of the pdbackup command are written to
the msg___pdbackup.log file in the current temporary directory.

Gathering WebSEAL information

To gather information about WebSEAL, use the /opt/pdweb/etc/pdinfo-pdweb.1st
service list file with the pdbackup command. For example:
pbackup -action backup -list /opt/pdweb/etc/pdinfo-pdweb.1st

On UNIX systems, this command invocation creates a file called
/var/PolicyDirector/pdbackup/pdinfo-pdweb.1st_ddmmyyyy.hh_mm.tar, where the
current date and time are substituted for ddmmyyyy and hh_mm, respectively.

Similarly, on Windows systems, the file would be called
%PD_HOME%\pdbackup\pdinfo-pdweb.1st_ddmmyyyy.hh_mm.dar. A representative file
name would be pdinfo-pdweb.1st_14Oct2003.11_22.dar.
Messages associated with the running of the `pdbbackup` command are written to the `msg__pdbbackup.log` file in the current temporary directory.

**Gathering Plug-in for Web Servers information**

To gather information about the Plug-in for Web Servers, use the `/opt/pdwebpi/etc/pdinfo-pdwebpi.lst` service list file with the `pdbbackup` command. For example:

```
 pdbbackup -action backup -list /opt/pdwebpi/etc/pdinfo-pdwebpi.lst
```

On UNIX systems, this command invocation creates a file called `/var/PolicyDirector/pdbackup/pdinfo-pdwebpi.lst_ddmmmyyyy.hh_mm.tar`, where the current date and time are substituted for `ddmmmyyyy` and `hh_mm`, respectively.

Similarly, on Windows systems, the file would be called `%PD_HOME%\pdbackup\pdinfo-pdwebpi.lst_ddmmmyyyy.hh_mm.tar`.

Messages associated with the running of the `pdbbackup` command are written to the `msg__pdbbackup.log` file in the current temporary directory.

**Validating and maintaining policy databases (pdacld_dump)**

The `pdacld_dump` command, located under the installation directory in the `sbin` subdirectory, is a serviceability utility used to validate and maintain Tivoli Access Manager policy (authorization) databases and their replicas. This command is installed as part of the policy server and is currently not translated.

**Note:** Some options to the `pdacld_dump` command require an understanding of the authorization database components and structure. It is recommended that you always work with the advice and assistance of Tivoli Customer Support when using this utility.

The name of the `pdacld_dump` utility does not accurately describe its function. The utility is not directly connected with the Tivoli Access Manager authorization server (`pdamgr`).

The `pdacld_dump` command examines the content and structure of a Tivoli Access Manager policy database. Tivoli Access Manager authorization databases include the master database (`master_authzn.db` – controlled by the policy server, `pdmgrd`) and replica databases. Replica databases are used by all instances of the Tivoli Access Manager authorization server (`pdacld`) and by any C language applications running in local mode. Each domain associated with Tivoli Access Manager also has a policy database associated with it, which might also have replica databases associated with it.

Some of the functions provided by the utility include:
- Transform the binary content of the specified database file into readable text (directed to standard output or redirected to a file)
- Produce a summary statement describing the condition of the specified database
- Examine the specified database for any corrupted content, defragment the database structure, and produce a valid updated version of the database
- Provide two levels of validation checking
Displaying the entire database

One of the primary uses of the `pdacld_dump` command is to obtain a readable display of the contents of a policy database. For example, to examine the policy database associated with the Zebra domain, issue the following command on a Windows system:

```
pdacld_dump -f "C:\Program Files\Tivoli\Policy Director\db\Zebra.db"
```

**Note:** On Microsoft Windows systems, the path and file name should be enclosed in double quotes as shown.

The following abbreviated example output has been provided to show representative content from the initial output entries and then the concluding summary section:

```
START OBJECT
Object type: 1291
Object size: 18
Object name: /auth/pobject/Management/Groups/ivmgrd-servers
Object seqnum: 0
IVPobj:
    Description: 
    Type: 0
    Is Policy Attachable: 1
END OBJECT

START OBJECT
Object type: 1291
Object size: 38
Object name: /auth/pobject/Management/POP
Object seqnum: 0
IVPobj:
    Description: POP Management root.
    Type: 16
    Is Policy Attachable: 1
END OBJECT

START OBJECT
Object type: 1282
Object size: 32
Object name: /auth/acl-map/default-config
Object seqnum: 0
ACL to Protected object map:
    ACL is attached at:
    /Management/Config
END OBJECT

START OBJECT
Object type: 1281
Object size: 26
Object name: /auth/pobject-map
Object seqnum: 0
Protected object to ACL map:
    Attached ACLs:
        default-root
END OBJECT

START OBJECT
Object type: 1291
Object size: 18
Object name: /auth/pobject/Management/Groups/SecurityGroup
Object seqnum: 0
IVPobj:
    Description:
```
Type: 0
Is Policy Attachable: 1
END OBJECT

START OBJECT
Object type: 1281
Object size: 25
Object name: /auth/pobject-map/Management/GSO
Object seqnum: 0
Protected object to ACL map:
  Attached ACLs:
    default-gso
END OBJECT

START OBJECT
Object type: 1291
Object size: 48
Object name: /auth/pobject/Management/Config
Object seqnum: 0
IVPobj:
  Description: Configuration Management root.
    Type: 16
    Is Policy Attachable: 1
END OBJECT

START OBJECT
Object type: 1293
Object size: 165
Object name: /auth/extended-acl/default-gso
Object seqnum: 0
ACL:
  Sequence number: 0
  Description: Default GSO ACL.
  Entries:
    ACLEntry: Group 81b1955c-00e2-11d8-9c4d-00096be9194c 0x205dc3
    ACLEntry: Cell 00000000-0000-0000-0000-000000000000 0x401
    ACLEntry: Unauthenticated 00000000-0000-0000-0000-000000000000 0x401
END OBJECT

START OBJECT
Object type: 1293
Object size: 179
Object name: /auth/extended-acl/default-management
Object seqnum: 0
ACL:
  Sequence number: 0
  Description: Default Management ACL.
  Entries:
    ACLEntry: Group 81b1955c-00e2-11d8-9c4d-00096be9194c 0xe07fc3
    ACLEntry: Group 819e828c-00e2-11d8-9c4d-00096be9194c 0x201
    ACLEntry: Cell 00000000-0000-0000-0000-000000000000 0x401
END OBJECT

START OBJECT
Object type: 1291
Object size: 18
Object name: /auth/pobject/Management/Groups/secmgrd-servers
Object seqnum: 0
IVPobj:
  Description:
    Type: 0
END OBJECT
Summary of a policy or replica database

The summary report from the pdacl_dump command reveals important information about the condition of a policy database or a replica. That information includes:

- The DB Sequence Number. The sequence number changes each time the database is updated. If you compare this number with the master authorization database sequence number, you can determine if the two databases are synchronized or not.
- The "Dumped" line reveals whether the file contains the complete database. It is possible for a database to be truncated, and therefore missing data.
• The "invalid objects were encountered" line indicates the validity of the database contents.

To obtain just the summary of a database, use the -s option of the pdacld_dump command. For example, to obtain just the summary of the policy database for the default management domain on a UNIX system:

```
pdacld_dump -f /var/PolicyDirector/db/master_authzn.db -s
```

Output similar to that shown in Figure 1 is written to standard output.

---

**Summary for ..\db\master_authzn.db**

Dumped 52 of 52 objects.

```
DB Sequence number : 51
DB SSL Sequence number : 1001
```

**FrequencyCount vs ObjectType vs BasePrefix summary**

- 1281:/auth/pobject-map
- 1282:/auth/acl-map
- 1283:/auth/dbinfo
- 1284:/auth/extern-auth
- 1285:/auth/actions
- 1287:/auth/pop
- 1288:/auth/pobject-popmap
- 1289:/auth/pop-map
- 1290:/auth/pobjectspace
- 1291:/auth/pobject
- 1292:/auth/extendattr
- 1293:/auth/extended-acl
- 1294:/auth/action-groups
- 1295:/auth/rule
- 1296:/auth/pobject-rulemap
- 1297:/auth/rule-map
- 1298: NULL objects
- 255: RESERVED-typed objects
- 0:: v37 objects
- 0:: unknown objects

0 invalid objects were encountered.

Database passes specified validation level check.

**Figure 1. Sample summary report from pdacld_dump**

**Repairing a damaged policy database**

In the event that the policy database has become damaged or corrupted, the pdacld_dump command also can be used to create a new policy database containing just the valid data recovered from the damaged policy database.

For example, if the policy database associated with the OutCenter domain has become damaged, use the following command to recover the valid information from the existing policy database and write it to a new policy database called RepairedOutCenter.db:

```
pdacld_dump -f /var/PolicyDirector/db/OutCenter.db \ 
   -r /var/PolicyDirector/db/RepairedOutCenter.db
```

Additionally, this option “defragments” the content of the new policy database.
To replace the damaged policy database with the repaired one, do the following:

1. Stop the policy or authorization server.
2. Rename the damaged policy database (OutCenter.db in the previous example) or move it to a different directory.
   
   ren OutCenter.db DamagedOutCenter.db

3. Rename the repaired file to have the same name as the original policy database.
   
   ren RepairedOutCenter.db OutCenter.db

4. Restart the policy or authorization server.
Chapter 3. Troubleshooting installation problems

This chapter describes the information recorded during the installation and configuration of IBM Tivoli Access Manager (Tivoli Access Manager) and its associated prerequisite products.

Problems with installation

Problems encountered with the easy installation program typically fall into one of two categories:

- Problems launching and running the InstallShield MultiPlatform installation program
- Problems installing or configuring the product.

When installing the product using the native installation tools, the InstallShield MultiPlatform installation program is only an issue on Microsoft Windows systems.

Problem with insufficient space for temporary files

The installation program might fail if there is insufficient disk space for temporary files. You can correct this problem by making more space available in the system’s temporary directory (typically /tmp or /var/tmp on UNIX systems, and %TEMP% on Windows systems).

Alternately, you can explicitly specify an alternate location using the –is:tempdir option when running the program. For example, to install the policy server on a UNIX system and specify a different location for temporary files created by the installation program:

```
install_ammgr -is:tempdir /var/scratch3
```

Problem connecting to display

The installation programs run in GUI mode. On UNIX systems, if the $DISPLAY environment variable is not set, the installation program fails with an error indicating that it cannot connect to the X11 window server. To correct the problem, set the $DISPLAY environment variable to a valid X11 server. In most cases, the following is sufficient, substituting the name of the system for mymachine.mycompany.com:

```
DISPLAY=mymachine.mycompany.com:0
```

Use the silent installation features to install without a GUI.

Removing the ibmjcaprovider.jar file

When installing the Tivoli Access Manager Java runtime environment component, the installation program might prompt you to remove the $JAVA_HOME/lib/ext/ibmjcaprovider.jar file and restart the installation program.

You must physically remove this file from the directory. Do not attempt to just rename the file, or to place the file in a subdirectory of the ext directory.

The JRE opens all files in this directory tree (regardless of name or extension) to determine what classes are available. The first file encountered by the JRE with a
specific class is the one that is used. However, the algorithm used to locate these files is platform and JRE specific, thus it can not easily be determined which file will be selected if multiple files exist in the directory tree with the requested class. Removing the existing ibmjcaprovider.jar file ensures that the proper classes are used by applications using the Tivoli Access Manager Java runtime environment.

**Problems running the installation program**

If the InstallShield MultiPlatform program itself fails before launching the installation panels or while you are completing the installation panels, additional diagnostic information can usually be obtained by enabling the Java console log and re-running the installation program. To display the Java console as a separate window, use the `-is:javaconsole` option:

```
install_amrte -is:javaconsole
```

You also can direct the Java console output directly to a file using the `-is:log` option:

```
install_amproxy -is:log "c:\AccessManagerLogs\amproxy_ismp.log"
```

Be sure to specify a valid location, as no error message is issued if the name is not valid, or if the JRE cannot write to the proposed file location.

**Problems installing and configuring the product**

Problems encountered after you have filled in the installation panels, reviewed the configuration options summary page, and clicked Next to begin the installation can usually be diagnosed by reviewing the installation log files created by the easy installation programs. These installation log files are written to the system’s temporary directory (typically `/tmp` or `/var/tmp` on UNIX systems, and `%TEMP%` on Windows systems).

Table 2 provides the names of the various installation log files created by the installation programs.

<table>
<thead>
<tr>
<th>Component</th>
<th>Installation log file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy server</td>
<td>msg__ammgr_install.log</td>
</tr>
<tr>
<td>Policy proxy server</td>
<td>msg__amproxy_install.log</td>
</tr>
<tr>
<td>Authorization server</td>
<td>msg__amacld_install.log</td>
</tr>
<tr>
<td>Runtime</td>
<td>msg__amrte_install.log</td>
</tr>
<tr>
<td>Java runtime</td>
<td>msg__amjrte_install.log</td>
</tr>
<tr>
<td>ADK</td>
<td>msg__amadk_install.log</td>
</tr>
<tr>
<td>Web Portal Manager</td>
<td>msg__amwpm_install.log</td>
</tr>
<tr>
<td>WebSEAL</td>
<td>msg__amweb_install.log</td>
</tr>
<tr>
<td>WebSEAL Application Development Kit</td>
<td>msg__amwebadk_install.log</td>
</tr>
<tr>
<td>Plug-in for Web Servers</td>
<td>msg__amwpismp_install.log</td>
</tr>
<tr>
<td>WebSphere Application Server and BEA WebLogic Server integration support</td>
<td>msg__amismp.log</td>
</tr>
<tr>
<td>Attribute retrieval service</td>
<td>msg__amars_install.log</td>
</tr>
</tbody>
</table>
Table 2. Installation log file names for easy installation programs (continued)

<table>
<thead>
<tr>
<th>Component</th>
<th>Installation log file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tivoli Identity Manager Provisioning Fast Start</td>
<td>msg__ampfs_install.log</td>
</tr>
<tr>
<td>IBM Tivoli Directory Server</td>
<td>msg__ldaps_install.log</td>
</tr>
</tbody>
</table>

Note: There are two underscore characters (__) after the characters msg in the log file names.

Native installation log files

Tivoli Access Manager native installation log files contain the completion status for the installation tasks performed. If the components are installed using the installation programs provided with the operating system, these are the only installation log files created. These native installation log files contain messages that are generated during the installation of the product.

Table 3. Native installation log files

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Native installation log file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>/smit.log</td>
</tr>
<tr>
<td>HP-UX</td>
<td>/var/adm/sw/swinstall.log</td>
</tr>
<tr>
<td>Linux</td>
<td>/tmp/install.log or /var/tmp/install.log</td>
</tr>
<tr>
<td>Solaris</td>
<td>Refer to the contents of the pkginfo files stored in the subdirectories of the /var/sadm/pkg directory</td>
</tr>
<tr>
<td>Windows</td>
<td>%PD_BASE%\log\msg__PDInstall.log</td>
</tr>
</tbody>
</table>

Note: There are two underscore characters (__) in the log file name.

WebSEAL has its own native installation log:
WebSEAL_install_dir\PDWeb_install.log

Native configuration log files

The Tivoli Access Manager pdconfig command is used to configure the Tivoli Access Manager Base product. Similar configuration commands are used for the Web Security components. Messages generated during the configuration process are stored within Tivoli Access Manager configuration log files.

Table 4. Default locations for configuration log files

<table>
<thead>
<tr>
<th>Component</th>
<th>Default configuration files on Microsoft Windows</th>
<th>Default configuration files on UNIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>%PD_BASE%\log\msg__config.log</td>
<td>none</td>
</tr>
<tr>
<td>Web Portal Manager</td>
<td>%PD_BASE%\log\msg__amwpmcfg.log</td>
<td>/var/PolicyDirector/log/msg__amwpmcfg.log</td>
</tr>
<tr>
<td></td>
<td>%PD_BASE%\log\amwpmcfg1.log</td>
<td>/var/PolicyDirector/log/amwpmcfg1.log</td>
</tr>
<tr>
<td>Java runtime environment</td>
<td>%PD_BASE%\log\msg__PDJrteCfg1.log</td>
<td>/var/PolicyDirector/log/msg__PDJrteCfg1.log</td>
</tr>
</tbody>
</table>
Table 4. Default locations for configuration log files (continued)

<table>
<thead>
<tr>
<th>Component</th>
<th>Default configuration files on Microsoft Windows</th>
<th>Default configuration files on UNIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSEAL</td>
<td>WebSEAL-install-dir\log\msg__amweb_config.log</td>
<td>/var/pdweb/log/msg__amweb_config.log</td>
</tr>
<tr>
<td>Plug-in for Web Servers</td>
<td>pdwebpi_install_dir\log\msg__pdwpicfg.log</td>
<td>/var/pdwebpi/log/msg__pdwpicfg.log</td>
</tr>
</tbody>
</table>
Chapter 4. Default message logging

This chapter describes the default message logging for runtime messages, Java runtime messages, and server messages produced by IBM Tivoli Access Manager (Tivoli Access Manager). The message log files associated with these different types of messages also are described.

There is no default tracing activity in Tivoli Access Manager. To enable tracing, see Chapter 6, “Trace logging,” on page 43.

Information on tailoring message logging to your needs or to aid in the diagnosis of a problem is described in Chapter 5, “Tailoring message logging,” on page 27. Unless otherwise noted, the information in this chapter describes the default message routing information.

Note: The logging of messages from the C language portions of the Tivoli Access Manager product is controlled through the use of routing files. Similarly, the logging of messages from the Java language portions of Tivoli Access Manager is controlled through the use of Java properties files. Distinctions between the two methods of message handling are mentioned when relevant.

Tivoli Message Standard message format

A message following the Tivoli Message Standard consists of a message identifier (ID) and message text. A message number is also associated with a message. A message number is a unique 32-bit value, displayed in decimal or hexadecimal, that some commands and APIs return to indicate that an operation was not successful. All messages adhering to the Tivoli Message Standard are listed in the IBM Tivoli Access Manager Error Message Reference with a detailed explanation and suggested action.

Message ID format

A message ID consists of 10 alphanumeric characters that uniquely identify the message. The message ID is composed of:

- a 3-character product identifier (see “Product identifiers” on page 2 for the list of identifiers used by Tivoli Access Manager)
- a 2-character component or subsystem identifier
- a 4-digit serial or message number
- a 1-character type code indicating the severity of the message

Figure 2 on page 20 shows the format of a message ID.
There are several different types of messages produced by Tivoli Access Manager:

**runtime messages**
These are messages generated by applications, commands, and utilities using the Tivoli Access Manager runtime component, as well as from C language-based Tivoli Access Manager components themselves, such as WebSEAL. They are written to the runtime message logs described in this chapter based on their severity level. These messages follow the Tivoli Message Standard and have an associated message ID. Additional information on these messages can be found in the *IBM Tivoli Access Manager Error Message Reference*.

**Java runtime messages**
These are messages generated by applications, commands, and utilities using the Tivoli Access Manager Java runtime environment, as well as Tivoli Access Manager Java language-based components. They are written to the Java runtime message logs described in this chapter. These messages tend to provide exception and stack trace information from the JRE.

**server messages**
These are messages generated by the daemons and servers associated with Tivoli Access Manager. Messages from the policy server, authorization server, WebSEAL servers, and policy proxy server are written to the server message logs described in this chapter. These messages also follow the Tivoli Message Standard and have an associated message ID. Additional information on these messages can be found in the *IBM Tivoli Access Manager Error Message Reference*.

**installation and configuration messages**
These are messages generated by the InstallShield MultiPlatform installation programs as well as by the configuration utilities. Some of these messages follow the Tivoli Message Standard and have an associated ID. They are written to the log files described in Chapter 3, “Troubleshooting installation problems,” on page 15 during installation and are not covered in this chapter.
WebSEAL HTTP messages
WebSEAL provides the capability of logging HTTP messages. This message logging capability is completely described in the IBM Tivoli Access Manager for e-business WebSEAL Administration Guide and is not covered in this guide.

Note: Tivoli Access Manager is written using both C and Java language programs and applications. Other applications written to use the APIs supplied by Tivoli Access Manager are also written in the C and Java languages.

Location of message log files
The location of the runtime and server message log files depends on whether Tivoli Common Directory support was requested during the installation of either the Tivoli Access Manager runtime component or the specific server. If the Tivoli Common Directory is not used, the message logs are located in the directories specified in Table 5.

Table 5. Location of message log files without Tivoli Common Directory

<table>
<thead>
<tr>
<th>Component</th>
<th>Default message log location on UNIX systems</th>
<th>Default message log location on Windows systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime message logs</td>
<td>/var.ibm/tivoli/common/HPD/logs</td>
<td>C:\Program Files\IBM\Tivoli\Common\HPD\logs</td>
</tr>
<tr>
<td>Policy server</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authorization server</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Policy proxy server</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebSEAL</td>
<td>$PD_WEB/log</td>
<td></td>
</tr>
<tr>
<td>Plug-in for Web Servers</td>
<td>$PD_WEBPI/log</td>
<td></td>
</tr>
<tr>
<td>Attribute retrieval service</td>
<td>$WAS_HOME</td>
<td></td>
</tr>
<tr>
<td>IBM WebSphere Edge Server</td>
<td></td>
<td></td>
</tr>
<tr>
<td>integration support</td>
<td>Windows edgepi_install_dir\cp\logs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UNIX /var.ibm/edgei\cp\server_root\logs</td>
<td></td>
</tr>
<tr>
<td>IBM WebSphere Application Server</td>
<td>$WAS_HOME/logs/node_name</td>
<td></td>
</tr>
<tr>
<td>integration support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BEA WebLogic Server</td>
<td>$BEA_HOME/user_projects/server_name</td>
<td></td>
</tr>
<tr>
<td>integration support</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If Tivoli Common Directory support is requested, the existing Tivoli Common Directory is used for storing most of the message logs. If no Tivoli Common Directory exists, one is created at the specified location. Tivoli Access Manager makes the default for the Tivoli Common Directory within the product installation directory. Table 6 shows the default location for the Tivoli Common Directory files assuming that the user has specified the recommended default value used by other Tivoli products.

Table 6. Location of message log files using the Tivoli Common Directory recommended defaults

<table>
<thead>
<tr>
<th>Component</th>
<th>Default message log location on UNIX systems</th>
<th>Default message log location on Windows systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime message logs</td>
<td>/var.ibm/tivoli/common/HPD/logs</td>
<td>C:\Program Files\IBM\Tivoli\Common\HPD\logs</td>
</tr>
<tr>
<td>Policy server</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authorization server</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Policy proxy server</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebSEAL</td>
<td>/var.ibm/tivoli/common/HPW/logs</td>
<td>C:\Program Files\IBM\Tivoli\Common\HPW\logs</td>
</tr>
</tbody>
</table>

Chapter 4. Default message logging
Table 6. Location of message log files using the Tivoli Common Directory recommended defaults (continued)

<table>
<thead>
<tr>
<th>Component</th>
<th>Default message log location on UNIX systems</th>
<th>Default message log location on Windows systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plug-in for Web Servers</td>
<td>/var.ibm/tivoli/common/AMZ/log</td>
<td>C:\Program Files\IBM\Tivoli\Common\AMZ\log</td>
</tr>
</tbody>
</table>

**Note:** Regardless of whether the Tivoli Common Directory is used, WebSphere Edge Server, WebSphere Application Server, and BEA WebLogic Server always write their messages to their respective application server log files as indicated in Table 5 on page 21.

### Message severity

A severity level is associated with each message in the runtime message log and the server message log. Table 7 provides a list of the severities along with their descriptions.

**Table 7. Message severity levels and associated message logs**

<table>
<thead>
<tr>
<th>Message severity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FATAL</td>
<td>An unrecoverable error has occurred. The process encountering the error usually terminates, and a core file might be produced. Special recovery action might be required. Depending on the nature of the failure, Customer support might need to be consulted.</td>
</tr>
<tr>
<td>ERROR</td>
<td>An unexpected, non-terminal, or correctable event has occurred. The product continues to function, but some services or functionality might not be available. Administrative action might be required.</td>
</tr>
<tr>
<td>WARNING</td>
<td>An event has occurred that is possibly not the desired or requested result. The program continues to function normally. The event might convey information useful for administrative action to avoid a potential error condition. For example, a low memory or disk space condition.</td>
</tr>
<tr>
<td>NOTICE</td>
<td>An event has taken place that does not directly require action. The event conveys general information about running state or normal actions.</td>
</tr>
<tr>
<td>NOTICE_VERBOSE</td>
<td>This is similar to NOTICE, but the event logged might contain more detailed information.</td>
</tr>
</tbody>
</table>

### Runtime messages and message logs

Tivoli Access Manager runtime messages are messages produced by applications, commands, and utilities using the Tivoli Access Manager runtime component. This includes the C language-based command line utilities, such as `pdadmin` and `svrsslcfg`.

Table 8 lists the names of the default message log files for both C and Java language applications.

**Table 8. Message severity levels and associated message logs**

<table>
<thead>
<tr>
<th>Message severity</th>
<th>Default runtime message log file name</th>
<th>Default Java runtime message log file names</th>
<th>Default WebSEAL message log file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>FATAL</td>
<td>msg__fatal.log</td>
<td>msg__app_nameN.log</td>
<td>STDERR</td>
</tr>
<tr>
<td>ERROR</td>
<td>msg__error.log</td>
<td>msg__app_nameN.log</td>
<td>STDERR</td>
</tr>
</tbody>
</table>
Table 8. Message severity levels and associated message logs (continued)

<table>
<thead>
<tr>
<th>Message severity</th>
<th>Default runtime message log file name</th>
<th>Default Java runtime message log file names</th>
<th>Default WebSEAL message log file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>WARNING</td>
<td>msg__warning.log</td>
<td>msg__app_nameN.log</td>
<td>STDERR</td>
</tr>
<tr>
<td>NOTICE</td>
<td>msg__notice.log</td>
<td>msg__app_nameN.log</td>
<td>msg__notice_PID.log</td>
</tr>
<tr>
<td></td>
<td>Note: Logging not enabled by default.</td>
<td></td>
<td>Note: Logging not enabled by default.</td>
</tr>
<tr>
<td>NOTICE_VERBOSE</td>
<td>msg__verbose.log</td>
<td>msg__app_nameN.log</td>
<td>msg__verbose_PID.log</td>
</tr>
<tr>
<td></td>
<td>Note: Logging not enabled by default.</td>
<td></td>
<td>Note: Logging not enabled by default.</td>
</tr>
</tbody>
</table>

Notes:

1. When WebSEAL is running as a background process, FATAL, ERROR, and WARNING messages are redirected to the server message log file associated with the WebSEAL instance: msg__webseal--instance_name.log.

2. If an application-specific configuration file does not exist for the Java application, message logging is controlled by the $JAVA_HOME/PolicyDirector/PDJLog.properties file. In that case, messages are written to the following files:

   - FATAL: msg__amj_fatalN.log
   - ERROR: msg__amj_errorN.log
   - WARNING: msg__amj_warningN.log
   - NOTICE: msg__amj_noticeN.log
   - NOTICE_VERBOSE: msg__amj_verboseN.log

   Note: Logging for NOTICE and NOTICE_VERBOSE messages not enabled by default.

Runtime messages for C language applications are written to different runtime message log files based on their severity. For example, WARNING messages generated from a C language application are written to the msg__warning.log file and FATAL messages are written to msg__fatal.log file. Error messages from WebSEAL are written to STDERR, unless WebSEAL is running in the background, in which case the messages are written to the WebSEAL server log.

Runtime message log files associated with C language applications are allowed to grow without bound. Periodically check the available disk space and make adjustments as necessary, perhaps by archiving or pruning the log files. You can change the name, location, and put size constraints on the runtime message log files, as explained in "Routing files" on page 27.

Runtime message log files associated with Java language applications can grow to a maximum size of 512 KB. A maximum of three message files can exist, with the most recent messages always being in the file ending in "1". When the file reaches its maximum size, the files are renamed. For example, when the msg__appname1.log file reaches 512 KB in size, the msg__appname3.log file is deleted (if it exists), the msg__appname2.log file (if it exists) is renamed msg__appname3.log, the msg__appname1.log file is renamed msg__appname2.log, and a new msg__appname1.log file is created.

Chapter 4. Default message logging 23
The names, location, number, and size of the Java runtime message logs can be changed, as explained in Table 12 on page 34.

**Server messages and message logs**

Server messages are messages generated by the daemons and servers associated with Tivoli Access Manager, such as the following:

- Policy server
- Authorization server
- WebSEAL server
- Policy proxy server
- Plug-in for Web Servers
- Plug-in for Edge Server
- WebSphere Application Server integration support
- BEA WebLogic Server integration support

Unlike runtime messages from C applications, which are written to different log files based on severity, server messages are always written to the message log associated with that particular server. Thus, all FATAL, WARNING, ERROR, NOTICE, and NOTICE_VERBOSE messages for the policy server are written to the `msg__pdmgrd_utf8.log` file. Similarly, WebSEAL messages are written to the `msg__webseald-instance_name.log` file.

Table 9 lists the default names for the server message log files.

<table>
<thead>
<tr>
<th>Server</th>
<th>Default message log file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy server</td>
<td><code>msg__pdmgrd_utf8.log</code></td>
</tr>
<tr>
<td>Authorization server</td>
<td><code>msg__pdacld_utf8.log</code></td>
</tr>
<tr>
<td>WebSEAL</td>
<td><code>msg__webseald-instance_name.log</code></td>
</tr>
<tr>
<td>Plug-in for Web Servers</td>
<td><code>msg__pdwebpi.log</code></td>
</tr>
<tr>
<td>Policy proxy server</td>
<td><code>msg__pdmgrproxyd_utf8.log</code></td>
</tr>
<tr>
<td>Attribute retrieval service</td>
<td><code>msg__amwebars_exceptions.log</code></td>
</tr>
<tr>
<td>IBM WebSphere Edge Server integration support</td>
<td><code>edge.date</code></td>
</tr>
<tr>
<td>IBM WebSphere Application Server integration support</td>
<td><code>SystemOut.log</code></td>
</tr>
<tr>
<td>BEA WebLogic Server integration support</td>
<td><code>server_name.log</code></td>
</tr>
</tbody>
</table>

**Note:** The messages associated with the integration support for WebSphere Edge Server, WebSphere Application Server, and BEA WebLogic Server are written to their respective application server log files. See the documentation associated with the application server for more information.

By default, the Tivoli Access Manager server message log files (the ones that start with `msg__`) are allowed to grow without bound. Be sure to periodically check the available disk space and make adjustments as necessary. You might want to archive or prune the log files on a periodic basis as well.
You can change the name, location, and put size constraints on the Tivoli Access Manager server message log files as explained in “Routing files” on page 27.

For the Tivoli Access Manager Plug-in for Web Servers component, message log entries are always written, by default, to the same set of files when the Tivoli Common Directory is not configured. These include:

**Windows:**

(log for the authorization server)

`webpi-install-dir\log\msg__pdwebpi.log`

(log for the IIS plug-in)

`webpi-install-dir\log\msg__pdwebpi-iis.log`

**UNIX:**

(log for the authorization server)

`/var/pdwebpi/log/msg__pdwebpi.log`

(log for the watchdog server)

`/var/pdwebpi/log/msg__pdwebpimgr.log`

### Message log entry format

**Figure 3** shows an example of a message log entry taken from a Tivoli Access Manager server message log file written in text format. The entry shown in the figure has been split into multiple lines for readability.

```
2003-10-26-20:09:10.984-06:00I----- 0x1354A41E pdmgrd ERROR
ivc socket e:\am510\src\mts\mtsclient.cpp 1832 0x000001c4
HPDCO1054E    Could not connect to the server acld2, on port 7137.
```

**Figure 3. Sample message log entry in text**

This message log entry consists of the following fields:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
<th>Value in sample entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>The time that the message entry was made.</td>
<td>2003-10-26-20:09:10.984-06:00I</td>
</tr>
<tr>
<td></td>
<td>This time is in the following format: CCYY-MM-DD–hh:mm:ss.fff[+</td>
<td>–]ii:jjI, where:</td>
</tr>
<tr>
<td></td>
<td>CCYY-MM-DD</td>
<td>Year, month, and day</td>
</tr>
<tr>
<td></td>
<td>hh:mm:ss.fff</td>
<td>Hours, minutes, seconds, and fractional seconds</td>
</tr>
<tr>
<td></td>
<td>ii:jjI</td>
<td>Time inaccuracy factor</td>
</tr>
<tr>
<td>message_number</td>
<td>32-bit message number, in hexadecimal</td>
<td>0x1354A41E</td>
</tr>
<tr>
<td>process_ID</td>
<td>Name of the process which created the entry</td>
<td>pdmgrd</td>
</tr>
<tr>
<td>severity_level</td>
<td>Severity of the message: FATAL, ERROR, WARNING, NOTICE, or NOTICE_VERBOSE</td>
<td>ERROR</td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
<td>Value in sample entry</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>component</td>
<td>Component associated with process that generated entry.</td>
<td>ivc</td>
</tr>
<tr>
<td>subcomponent</td>
<td>Subcomponent associated with process that generated entry</td>
<td>socket</td>
</tr>
<tr>
<td>src_file</td>
<td>Name of the product source file that generated the entry</td>
<td>e:\am510\src\mts\mtsclient.cpp</td>
</tr>
<tr>
<td>src_line</td>
<td>Line number in source file</td>
<td>1832</td>
</tr>
<tr>
<td>thread_id</td>
<td>Thread ID, in hexadecimal</td>
<td>0x000001c4</td>
</tr>
<tr>
<td>message_id</td>
<td>Tivoli message identifier</td>
<td>HPDCO1054E</td>
</tr>
<tr>
<td>text</td>
<td>Message text</td>
<td>Could not connect to the server acld2, on port 7137.</td>
</tr>
</tbody>
</table>

Similarly, if the same message log entry was written in Tivoli XML Log format, it would look as shown in Figure 4.

```xml
<Message Id="HPDCO1054E" Severity="ERROR">
  <Time Millis="1067220550984">2003-10-26-20:09:10.984</Time>
  <Component>ivc/socket</Component>
  <LogAttribs><KeyName><![CDATA[Message Number]]></KeyName><Value><![CDATA[0x1354A41E]]></Value></LogAttribs>
  <Source FileName="e:\am510\src\mts\mtsclient.cpp" Method="unknown" Line="1832"></Source>
  <Process>pdmgrd</Process>
  <Thread>0x0000001c4</Thread>
  <TranslationInfo Type="XPG4" Catalog="pdbivc.cat" SetId="1" MsgKey="1354a41e">
    <Param><![CDATA[acld2]]></Param>
    <Param><![CDATA[7137]]></Param>
  </TranslationInfo>
  <LogText><![CDATA[HPDCO1054E Could not connect to the server acld2, on port 7137.]]></LogText>
</Message>
```

Figure 4. Sample message entry in Tivoli XML format

To configure Tivoli Access Manager message logging and trace logging to produce output in Tivoli XML Log format, see “Logging messages in Tivoli XML Log format” on page 31 and “Trace logging in Tivoli XML Log format” on page 47, respectively.

See “Tivoli XML Log Viewer” on page 39 for information on viewing message logs written in Tivoli XML Log format.
Chapter 5. Tailoring message logging

The message logging behavior of IBM Tivoli Access Manager (Tivoli Access Manager) is customizable. This chapter describes the use of routing files, Java properties files, and environment variables to control message logging. Entries in these files determine the types of messages that are logged and where and how those messages are recorded.

**Note:** The logging of messages from the C language portions of the Tivoli Access Manager product is controlled through the use of routing files. Similarly, the logging of messages from the Java language portions of Tivoli Access Manager is controlled through the use of Java properties files.

This chapter describes the following topics:
- “Routing files”
- “Limiting message log size with routing files” on page 32
- “Environment variables” on page 33
- “Java properties files” on page 34
- “Tailoring message logging in Web Portal Manager” on page 36
- “Tailoring message logging in WebSphere Application Server integration support” on page 37
- “Tailoring message logging in BEA WebLogic Server integration support” on page 38
- “Tailoring message logging in WebSphere Edge Server” on page 39

Routing files

Routing files are ASCII files that are used to customize the message logging and tracing associated with C language based Tivoli Access Manager servers, daemons, and other C language programs and applications. Information on enabling tracing can be found in Chapter 6, “Trace logging,” on page 43.

The contents of the routing file enable the user to control the following aspects of message logging:
- Whether message logging is on or off for each class of messages (FATAL, ERROR, WARNING, NOTICE, or NOTICE_VERBOSE)
- Where the message log output for each class of messages is to be directed
- If message output is being directed to a file, how many files for each class of messages should be used, and how many messages should be placed in each file


Location of routing files

The default locations for the routing files used by Tivoli Access Manager are shown in Table 11 on page 28.
Table 11. Routing files associated with Tivoli Access Manager

<table>
<thead>
<tr>
<th>Component</th>
<th>Default name and location of routing file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime environment</td>
<td>Windows %PD_HOME%etc\routing</td>
</tr>
<tr>
<td></td>
<td>UNIX /opt/PolicyDirector/etc/routing</td>
</tr>
<tr>
<td>Policy server</td>
<td>Windows %PD_HOME%etc\pdmgrd_routing</td>
</tr>
<tr>
<td></td>
<td>UNIX /opt/PolicyDirector/etc/pdmgrd_routing</td>
</tr>
<tr>
<td>Authorization server</td>
<td>Windows %PD_HOME%etc\pdacld_routing</td>
</tr>
<tr>
<td></td>
<td>UNIX /opt/PolicyDirector/etc/pdacld_routing</td>
</tr>
<tr>
<td>Policy proxy server</td>
<td>Windows %PD_HOME%etc\pdmgrproxyd_routing</td>
</tr>
<tr>
<td></td>
<td>UNIX /opt/PolicyDirector/etc/pdmgrproxyd_routing</td>
</tr>
<tr>
<td>WebSEAL server</td>
<td>Windows %PD_WEB%etc\routing</td>
</tr>
<tr>
<td></td>
<td>UNIX /opt/pdweb/etc/routing</td>
</tr>
</tbody>
</table>

Note: The Tivoli Access Manager Plug-in for Web Servers component sets the information that is typically contained within a routing file programmatically, therefore, it has no routing file of its own. The message logging behavior of the Plug-in for Web Servers can be altered through the use of the $PD_SVC_ROUTING_FILE environment variable.

The default routing file can be overwritten through the use of the $PD_SVC_ROUTING_FILE environment variable. If the fully qualified file name indicated by $PD_SVC_ROUTING_FILE does not exist or is not accessible, the default routing file shown in Table 11 is used.

Routing file message logging entry

The format of a routing file entry that controls message logging is:

```
severity:destination:location [[;destination:location] ...] [:GOESTO:other_severity]
```

where:

`severity`

Specifies the severity of messages to be logged.

Valid message severities are: FATAL, ERROR, WARNING, NOTICE, or NOTICE_VERBOSE.

An asterisk (*) can be used to specify all message severities.

`destination`

Specifies where the message log output for this class of messages should be directed. Valid destinations are:

`STDOUT`

Messages are written to the standard output device as ASCII text in the current code page and locale.
XMLSTDOUT
Messages are written to the standard output device in Tivoli Log XML format.

STDOUT
Messages are written to the standard error device as ASCII text in the current code page and locale.

XMLSTDERR
Messages are written to the standard error device in Tivoli Log XML format.

FILE
Entries are written to the specified file as ASCII text in the current code page and locale.

TEXTFILE
Same as FILE.

UTF8FILE
Messages are written to the specified file as UTF-8 text.

XMLFILE
Messages are written to the specified file in Tivoli Log XML format.

DISCARD
Messages are discarded.

If a destination of FILE, TEXTFILE, UTF8FILE, or XMLFILE is specified, it can optionally be followed by a period and two numbers separated by a period. These values indicate the number of files to be used and the number of message entries to be written to each file, respectively. If these values are omitted, one file is used and that file grows without limit.

location
Specifies the location for the message log data.

When destination is STDOUT, STDERR, XMLSTDOUT, XMLSTDERR, or DISCARD, a hyphen (–) must be specified for location.

When destination is FILE, TEXTFILE, UTF8FILE, or XMLFILE, location is the fully qualified file name where the messages are written. The character string $ld can be used to insert the process ID into the file name.

When the number of files and the number of messages per file are specified as part of the destination, a period and the file number are added to the end of the file name specified.

On UNIX systems, the file name specification is followed by the desired file permissions, the user that owns the file, and the group that owns the file, each separated by a colon (:).

Note: On Windows systems, the file name must not end with a period when more than one file is used, because this results in a file name containing two consecutive periods, which is not valid.

GOESTO other_severity | comp
Specifies that messages should additionally be routed to the same destinations and locations as either messages of other_severity or trace entries from component comp. This must be the last element in the message logging specification.
Lines in the file that start with a pound sign (#) are considered comments and do not affect logging.

**Understanding routing files by examining the defaults**

To understand how the default message logging (which is described in Chapter 4, “Default message logging,” on page 19) works, some default routing files are examined.

**Default Windows routing file**

The default %PD_HOME\etc\routing file on a Windows system contains message logging specifications similar to the one shown in Figure 5.

```
FATAL:STDERR:-;FILE:C:/PROGRA~1/Tivoli/POLICY~1/log/msg__fatal.log
ERROR:STDERR:-;FILE:C:/PROGRA~1/Tivoli/POLICY~1/log/msg__error.log
WARNING:STDERR:-;FILE:C:/PROGRA~1/Tivoli/POLICY~1/log/msg__warning.log
NOTICE:FILE:C:/PROGRA~1/Tivoli/POLICY~1/log/msg__notice.log
#NOTICE_VERBOSE:STDERR:-;FILE:C:/PROGRA~1/Tivoli/POLICY~1/log/msg__verbose.log
```

*Figure 5. Sample Windows routing file*

Using this routing file, FATAL messages are sent to the standard error device as ASCII text in the current code page and locale, and also are directed to the file C:/PROGRA~1/Tivoli/POLICY~1/log/msg__fatal.log in the same format. However, NOTICE messages are only written to the file C:/PROGRA~1/Tivoli/POLICY~1/log/msg__notice.log.

No action is taken on NOTICE_VERBOSE messages. To enable these messages to be sent to the standard error device and the file specified, remove the pound sign (#) from the beginning of the line and restart the application, server, or daemon.

**Default UNIX policy server routing file**

The default routing file for the policy server on a UNIX system, /opt/PolicyDirector/etc/pdmgrd_routing, contains message logging specifications similar to those in Figure 6.

```
FATAL:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdmgrd_utf8.log:644:ivmgr:ivmgr
ERROR:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdmgrd_utf8.log:644:ivmgr:ivmgr
WARNING:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdmgrd_utf8.log:644:ivmgr:ivmgr
NOTICE:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdmgrd_utf8.log:644:ivmgr:ivmgr
#NOTICE_VERBOSE:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdmgrd_utf8.log:644:ivmgr:ivmgr
```

*Figure 6. Sample policy server routing file*

With this specification, FATAL, ERROR, WARNING, and NOTICE messages are written to the standard output device as ASCII text in the current code page and locale, and all these messages are written as UTF-8 text to the file /var/PolicyDirector/log/msg__pdmgrd_utf8.log. When the log file is initially created, user ivmgr is assigned as the owner, the owning group is ivmgr, and the file permissions are set to 644. No action is taken on NOTICE_VERBOSE messages.

**Default UNIX WebSEAL routing file**

A portion of the default WebSEAL routing file on a UNIX system is shown in Figure 7 on page 31. Note the use of the pound sign (#) to add comments to the file.
By default WebSEAL writes all messages to standard error. When WebSEAL is running in the background, standard error is redirected to msg_webseald-<instance>.log.

`FATAL:STDERR:-
ERROR:STDERR:-
WARNING:STDERR:-`

If FILE ends with ".n.m" then at most "n" files of "m" entries each will be written where "n" will be appended to each generation of file. So to keep the last 1000 warnings around in 10 files for all programs:

`#NOTICE:FILE.10.100:/var/pdweb/log/msg__notice_%ld.log:644:ivmgr:ivmgr`  
`#NOTICE_VERBOSE:FILE.10.100:/var/pdweb/log/msg__verbose_%ld.log:644:ivmgr:ivmgr`

Figure 7. Sample WebSEAL routing file

By removing the pound sign (#) from the front of the NOTICE message logging specification, and then stopping and restarting the WebSEAL server, NOTICE messages are written to a set of 10 files. Assuming the process ID of the WebSEAL server is 1017, the names of those 10 files would be:

```
/var/pdweb/log/msg__notice_1017.log.1  
/var/pdweb/log/msg__notice_1017.log.2  
/var/pdweb/log/msg__notice_1017.log.3  
/var/pdweb/log/msg__notice_1017.log.4  
/var/pdweb/log/msg__notice_1017.log.5  
/var/pdweb/log/msg__notice_1017.log.6  
/var/pdweb/log/msg__notice_1017.log.7  
/var/pdweb/log/msg__notice_1017.log.8  
/var/pdweb/log/msg__notice_1017.log.9  
/var/pdweb/log/msg__notice_1017.log.10
```

Message logging starts with the first file, `/var/pdweb/log/msg__notice_1017.log.1`. After 100 NOTICE messages have been logged to that file, messages are written to the second file, `/var/pdweb/log/msg__notice_1017.log.2`. Message logging continues in this manner until 100 messages are written to the `/var/pdweb/log/msg__notice_1017.log.10` file. At that point, the messages in the first file are deleted and logging resumes again to the `/var/pdweb/log/msg__notice_1017.log.1` file.

Routing file examples

To illustrate other features available with the routing files, several additional examples are provided.

Logging messages in Tivoli XML Log format

To send FATAL, ERROR, and WARNING messages from the authorization server to the standard output device, to a file as UTF-8 text, and to another file in Tivoli Log XML format, the `%PD_HOME\etc\msg_pdacld.routing` file could be modified as follows:

```
FATAL:STDOUT:-;UTF8FILE:C:\LOGS\MSG__PDACLDB:\%LD.LOG;XMLFILE:C:\XMLLOGS\MSG__PDACLDB:\%LD.XML  
ERROR:STDOUT:-;UTF8FILE:C:\LOGS\MSG__PDACLDB:\%LD.LOG;XMLFILE:C:\XMLLOGS\MSG__PDACLDB:\%LD.XML  
WARNING:STDOUT:-;UTF8FILE:C:\LOGS\MSG__PDACLDB:\%LD.LOG;XMLFILE:C:\XMLLOGS\MSG__PDACLDB:\%LD.XML  
NOTICE:UTF8FILE:C:\LOGS\MSG__PDACLDB:\%LD.LOG;XMLFILE:C:\XMLLOGS\MSG__PDACLDB:\%LD.XML  
NOTICE_VERBOSE:UTF8FILE:C:\LOGS\MSG__PDACLDB:\%LD.LOG;XMLFILE:C:\XMLLOGS\MSG__PDACLDB:\%LD.XML
```

Figure 8. Routing file example showing Tivoli XML Log format
After stopping and restarting the authorization server to pick up the routing file change, and assuming that the process ID of the authorization server is 1253, two files are created containing messages:

C:\PDACLD_LOGS\MSG_PDAACL_1253.LOG  
C:\PDACLD_XMLLOGS\MSG_PDAACL_1253.XML

The Tivoli Log XML Viewer can be used to analyze the XML file produced. See "Tivoli XML Log Viewer" on page 39 for instructions on installing and using the viewer.

**Handling all messages the same way**

To send all runtime messages to a single file regardless of severity, the routing specification shown in Figure 9 could be used in the /opt/PolicyDirector/etc/routing file.

*:*UTF8FILE:/tmp/msg__amrte_utf8.log:666:ivmgr:ivmgr

**Figure 9. Sending all messages to one location**

**Handling some messages the same way**

To send all policy server messages to a single file, and also send FATAL and ERROR messages to the standard error device, the /opt/PolicyDirector/etc/pdmgrd_routing file could be modified as shown in Figure 10.

FATAL:STDERR:-;GOESTO:NOTICE_VERBOSE  
ERROR:STDERR:-;GOESTO:NOTICE_VERBOSE  
WARNING:GOESTO:NOTICE_VERBOSE  
NOTICE:GOESTO:NOTICE_VERBOSE  
NOTICE_VERBOSE:UTF8FILE:/tmp/msg__pdmgrd_utf8.log:644:ivmgr:ivmgr

**Figure 10. Using GOESTO**

FATAL messages are send to STDERR, and then to the same destinations as NOTICE_VERBOSE messages, namely written as UTF-8 text to the file /tmp/msg__pdmgrd_utf8.log.

**Limiting message log size with routing files**

By default, message logs grow without limit. This requires that the directories and file systems containing message log files be checked on a periodic basis to ensure that enough space is available, and to prune the log files or make additional space available as necessary.

The routing files can be modified to limit the amount of disk space used for message logging.

Consider the routing specification shown in Figure 11.

FATAL:STDOUT:-;UTF8FILE.10.100:/var/PolicyDirector/log/msg__pdmgrd_fatal_utf8.log:644:ivmgr:ivmgr  
ERROR:STDOUT:-;UTF8FILE.10.100:/var/PolicyDirector/log/msg__pdmgrd_error_utf8.log:644:ivmgr:ivmgr  
WARNING:STDOUT:-;UTF8FILE.5.1000:/var/PolicyDirector/log/msg__pdmgrd_warning_utf8.log:644:ivmgr:ivmgr  
NOTICE:STDOUT:-;UTF8FILE.5.1000:/var/PolicyDirector/log/msg__pdmgrd_notice_utf8.log:644:ivmgr:ivmgr  
NOTICE_VERBOSE:STDOUT:-;XMLFILE.10.500:/var/PolicyDirector/log/msg__pdmgrd_verbose_utf8.xml:644:ivmgr:ivmgr

**Figure 11. Multiple log files**
All message files produced by this message logging specification are of a determinate size, thus the maximum disk space that could be used by all of the files can be calculated.

**Determining maximum size of a message log**

Each entry made to a message log file created using a destination of FILE, TEXTFILE, or UTF8FILE is an average of 200 bytes in size. Therefore, the maximum size of a log file, in bytes, can be estimated as follows:

\[ 200 \times \text{(Number of log files)} \times \text{(Number of entries per log file)} \]

For example, given a specification of:

```
NOTICE:UTF8FILE.10.1000:E:\LOGS\PDPROXYMGRD.LOG
```

The maximum size for the PDPROXYMGRD.LOG file would be approximately \((200 \times 10 \times 1000)\) or 2,000,000 bytes.

Message log entries written in Tivoli XML Log format are an average of 650 bytes in size, thus for a specification of:

```
NOTICE:XMLFILE.10.500:E:\LOGS\MSG__NOTICE.XML
```

The maximum size would be approximately \((650 \times 10 \times 500)\) or 3,250,000 bytes.

**Environment variables**

The message logging behavior specified by a routing file can be changed through the use of environment variables. The `PD_SVC_ROUTING_FILE` environment variable can specify a fully qualified file name for a routing file to replace the one currently in use. If the file is not accessible, or does not exist, no change in message logging is made.

Routing for messages of a specific severity can be manipulated through the use of environment variables as well. Set the appropriate message log entry format string to the desired environment variable:

- `SVC_FATAL`
- `SVC_ERROR`
- `SVC_WARNING`
- `SVC_NOTICE`
- `SVC_NOTICE_VERBOSE`

For example, the following command on Windows overrides the setting in the corresponding routing file and directs WARNING messages to the standard error device and a file:

```
SET SVC_WARNING="STDERR:-;FILE:D:\MSGS\MSG__WARNING.LOG"
```

See “Routing file message logging entry” on page 28 for a description of message log entry format strings.
Java properties files

Java properties files are ASCII files that are used to customize the message logging and tracing associated with Java language based Tivoli Access Manager servers, daemons, and other Java language programs and applications. Information on enabling tracing can be found in Chapter 6, “Trace logging,” on page 43. These properties files also are used to configure other aspects of the application.

The contents of the properties file enables the user to control the following aspects of message logging:

- Whether message logging is on or off for each class of messages (FATAL, ERROR, WARNING, NOTICE, or NOTICE_VERBOSE)
- Where the message log output should be directed
- How many message logs should be used
- How large each message log file can be

Lines in the file that start with a pound sign (#) are comments and do not affect logging.

The application name (app_name) is part of the name of each logging property associated with a Java application. The application name is specified when using the com.tivoli.pd.jcfg.SvrSslCfg command.

See Appendix B, “Sample Java application properties file,” on page 111 for a sample of an application specific Java properties file.

Location of Java properties files

The default locations for the Java properties files used by Tivoli Access Manager are shown in Table 12.

<table>
<thead>
<tr>
<th>Component</th>
<th>Default name and location of Java properties file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java application configured using the com.tivoli.pd.jcfg.SvrSslCfg command</td>
<td>The output application configuration file as specified in the com.tivoli.pd.jcfg.SvrSslCfg command.</td>
</tr>
<tr>
<td>Java language based Tivoli Access Manager commands, such as pdjrtecfg and com.tivoli.pd.jcfg.SvrSslCfg or applications not explicitly configured.</td>
<td>$JAVA_HOME/PolicyDirector/PDJLog.properties</td>
</tr>
</tbody>
</table>
| Web Portal Manager | Windows %PD_HOME%\java\export\pdwpm\pdwpm.properties
UNIX /opt/PolicyDirector/java/export/pdwpm/pdwpm.properties |
| WebSphere Application Server integration support | $PDWAS_HOME/etc/jlog.properties |
| BEA WebLogic Server integration support | $JAVA_HOME/amwls/wls_domain_name/amwlsjlog.properties |

Note: If the com.tivoli.pd.jcfg.SvrSslCfg command has not been run, then no application-specific configuration file exists and the PDJLog.properties file is used.

See Appendix B, “Sample Java application properties file,” on page 111 for an annotated example of an application-specific properties file.
Message loggers and file handlers

Each properties file contains properties for one or more message loggers. The isLogging property specifies whether message logging is enabled or not. To turn logging on for a specific message logger, use:

```
baseGroup.PDJapp_nameMessageLogger.isLogging=true
```

To disable logging for a specific message logger, use:

```
baseGroup.PDJapp_nameMessageLogger.isLogging=false
```

Associated with each message logger is one or more file handlers. A file handler specifies the destination for messages. Once message logging is enabled by the message logger, the file handler properties are examined to determine if messages should be logged and, if so, how and where. The properties associated with a file handler are:

```
baseGroup.PDJapp_nameFileHandler.fileName=
baseGroup.PDJapp_nameFileHandler.maxFileSize=
baseGroup.PDJapp_nameFileHandler.maxFiles=
```

where:

- **fileName** Specifies the fully qualified file name to be used as the base name for message log files. The file can be in any location accessible by the Java application.

- **maxFileSize** Specifies the maximum size, in KB, of each message log file. Default is 512.

- **maxFiles** Specifies the maximum number of files to be used for message logging. Default is 3.

To specify what classes of messages should be logged, use the MessageAllMaskFilter.mask property as illustrated in Figure 12.

```
baseGroup.PDJapp_nameMessageAllMaskFilter.mask=FATAL | ERROR | WARNING | NOTICE | NOTICE_VERBOSE
```

Figure 12. Specifying what messages to log in a properties file

When PDJLog.properties is used

The $JAVA_HOME/PolicyDirector/PDJLog.properties file is used to define message and trace logging properties only in the following cases:

- For non application-related Java commands, such as pdjrtcfg and com.tivoli.pdj.cfg.SvrSslCfg.
- If a Java application has not been explicitly configured with the com.tivoli.pdj.cfg.SvrSslCfg command.
- If the application-specific properties file is inaccessible or does not exist.
- If a required property in the application-specific properties file is not found.

When using the default PDJLog.properties file, message logging is enabled only for FATAL, ERROR, and WARNING messages. This is shown in the portion of the PDJLog.properties file in Figure 13 on page 36. Logging can be enabled for NOTICE and NOTICE_VERBOSE messages by changing the isLogging property to true for the last two properties shown in the figure.
On a UNIX system, to enable both NOTICE and NOTICE_VERBOSE messages, as well as to change the destination properties of NOTICE_VERBOSE messages, the following changes can be made, as indicated in **bold**:

```plaintext
baseGroup.PDJMessageLogger.isLogging=true
baseGroup.PDJFatalFileHandler.isLogging=true
baseGroup.PDJErrorFileHandler.isLogging=true
baseGroup.PDJWarningFileHandler.isLogging=true
baseGroup.PDJNoticeFileHandler.isLogging=false
baseGroup.PDJNoticeVerboseFileHandler.isLogging=false
```

Figure 13. Portion of the default PDJLog.properties file

On a UNIX system, to enable both NOTICE and NOTICE_VERBOSE messages, as well as to change the destination properties of NOTICE_VERBOSE messages, the following changes can be made, as indicated in **bold**:

```plaintext
baseGroup.PDJNoticeFileHandler.isLogging=true
```

```plaintext
baseGroup.PDJNoticeVerboseFileHandler.fileName=/tmp/logs/msg__amjrte_verbose.log
baseGroup.PDJNoticeVerboseFileHandler.maxFileSize=1024
baseGroup.PDJNoticeVerboseFileHandler.maxFiles=4
baseGroup.PDJNoticeVerboseFileHandler.isLogging=true
```

After making these changes, NOTICE_VERBOSE messages are written to the `/tmp/logs/msg__amjrte_verbose.log1` file. After that file reaches 1024 KB in size, the file is renamed `/tmp/logs/msg__amjrte_verbose.log2` and logging continues with a new `/tmp/logs/msg__amjrte_verbose.log1` message log file. A maximum of 4 message log files are used.

(The procedure would be the same on a Windows system. The file name just needs to be changed to reflect a fully qualified file name on Windows.)


**Console handler and console message logging**

A console handler and a message console handler also are configured in the `$JAVA_HOME/PolicyDirector/PDJLog.properties` file. Both are disabled by default. To send messages to the console, set both `isLogging` properties to `true`:

```plaintext
baseGroup.PDJConsoleHandler.isLogging=true
baseGroup.PDJMessageConsoleHandler.isLogging=true
```

**Tailoring message logging in Web Portal Manager**

**Note:** Only change entries in this properties file at the request of IBM Customer Support for Tivoli software.

The properties file associated with Web Portal Manager is `pdwpm.properties`.

**Windows**

`%PD_HOME%\java\export\pdwpm\pdwpm.properties`

**UNIX**

`/opt/PolicyDirector/java/export/pdwpm/pdwpm.properties`

Message logging is enabled by default:

```plaintext
baseGroup.PDJamwpm-host_nameMessageLogger.isLogging=true
```
Tailoring message logging in WebSphere Application Server integration support

The properties file associated with Tivoli Access Manager WebSphere Application Server integration support is $PDWAS_HOME/etc/jlog.properties. Message logging is enabled by default. To disable message logging for all message loggers (not recommended):

baseGroup.MessageLogger.isLogging=false

Message logging for other message loggers can be selectively enabled using:

baseGroup.message_logger.isLogging=true

and disabled using:

baseGroup.message_logger.isLogging=false

If the isLogging property is not set for a specific trace logger, the setting of baseGroup.MessageLogger.isLogging is used.

The available message loggers are:

<table>
<thead>
<tr>
<th>message_logger</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMWASWebMessageLogger</td>
<td>Enables messages for the WebSphere Application Server authorization plug-in.</td>
</tr>
<tr>
<td></td>
<td>To diagnose most problems, usually only this message logger needs to be enabled.</td>
</tr>
<tr>
<td>AmasRBPFMessageLogger</td>
<td>Enables messages for the Role Based Policy Framework. This is the underlying framework used by the WebSphere Application Server integration support to make access decisions.</td>
</tr>
<tr>
<td>AmasCacheMessageLogger</td>
<td>Enables messages for the policy caches used by the Role Based Policy Framework.</td>
</tr>
</tbody>
</table>

After changing the properties file, enable the change in the WebSphere Application Server console:

**Version 5**

1. Click **Servers** \(\rightarrow\) **Application Servers** in the left frame.
2. Click on the desired server.
3. Click **Logging and Tracing** \(\rightarrow\) **Diagnostic Trace**.
4. Under the Trace Specifications heading, click **Modify**.
5. Click **Components** \(\rightarrow\) **PDWAS** and select the desired trace level.
6. Click **Apply**.

**Version 4**

1. Click **Servers** \(\rightarrow\) **Application Servers** in the left frame.
2. Click on the desired server.
3. Click **Logging and Tracing Properties** \(\rightarrow\) **Diagnostic Trace Service**.
4. In the **Trace Specification** box, enter the following:
   
   com.ibm.ws.security.PDWAS=all=enabled

5. Click **Apply**.
Save the configuration change, and then restart the server for this change to take effect.

As mentioned in Chapter 4, “Default message logging,” on page 19, messages are written to the SystemOut.log file associated with the WebSphere Application Server.

**Tailoring message logging in BEA WebLogic Server integration support**

The properties file associated with Tivoli Access Manager BEA WebLogic Server integration support is:

```
$JAVA_HOME/amwls/wls_domain_name/wls_realm_name/amwlsjlog.properties
```

where *wls_domain_name* is the name of your configured BEA WebLogic Server domain, and *wls_realm_name* is the name of your configured BEA WebLogic Server realm within the domain. Message logging is enabled by default. To disable message logging for all message loggers (*not recommended*):

```
baseGroup.MessageLogger.isLogging=false
```

Message logging for other message loggers can be selectively enabled using:

```
baseGroup.message_logger.isLogging=true
```

and disabled using:

```
baseGroup.message_logger.isLogging=false
```

If the *isLogging* property is not set for a specific trace logger, the setting of `baseGroup.MessageLogger.isLogging` is used.

The available message loggers are:

<table>
<thead>
<tr>
<th><code>message_logger</code></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AmasRBPFMessageLogger</strong></td>
<td>Enables messages for the Role Based Policy Framework. This is the underlying framework used by the BEA WebLogic Server integration support to make access decisions.</td>
</tr>
<tr>
<td><strong>AmasCacheMessageLogger</strong></td>
<td>Enables messages for the policy caches used by the Role Based Policy Framework.</td>
</tr>
<tr>
<td><strong>AMSSPICfgMessageLogger</strong></td>
<td>Enables messages for the configuration components of the BEA WebLogic Server integration support.</td>
</tr>
<tr>
<td><strong>AMSSPIAuthzMessageLogger</strong></td>
<td>Enables messages for the authorization components of the BEA WebLogic Server integration support.</td>
</tr>
<tr>
<td><strong>AMSSPIAuthnMessageLogger</strong></td>
<td>Enables messages for the authentication components of the BEA WebLogic Server integration support.</td>
</tr>
<tr>
<td><strong>AMSSPIRoleMapperMessageLogger</strong></td>
<td>Enables messages for programmatic role checks.</td>
</tr>
</tbody>
</table>
AMSSPIResourceManagerMessageLogger

Enables messages associated with the interfaces between BEA WebLogic Server and the Role Based Policy Framework.

After changing the properties file, restart the BEA WebLogic Server to pick up the change.

As mentioned in Chapter 4, “Default message logging,” on page 19, messages are written to the log file associated with the BEA WebLogic Server.

Tailoring message logging in WebSphere Edge Server

Tivoli Access Manager Plug-in for Edge Server does not make use of a routing file. The settings for message logging are defined within the object space definition configuration file, osdef.conf. The default location for this file is:

**Windows**

`edgepi_install_dir\etc`

**UNIX**

`/var.ibm/edge/etc`

Relevant sections of the osdef.conf file are:

```
# Indicates the category of logging messages that the plug-in will send to
# the event log file. Only messages matching the specified category will
# be sent to the log file. This option is valid only in the [Global]
# section of this file. The default value is 'EWI'.

# Log message flag options:
# 'E' - Error messages
# 'W' - Warning messages
# 'I' - Informational messages
# 'D' - Debugging messages
#logging_flags = EWI

# Indicates the verbosity level for informational and debugging log
# messages. This value can range from 0 to 5. A higher number means
# that more messages will be logged. This option is valid only in the
# [Global] section of this file. The default value is '3'.
#logging_level = 3
```

---

**Tivoli XML Log Viewer**

The Tivoli XML Log Viewer is used to process the Log XML files generated by the C language based components of Tivoli Access Manager and other Tivoli products. The viewer must be explicitly installed, as it is not included as part of any other Tivoli Access Manager component.

**Note:** Java language based Tivoli Access Manager components and applications can not produce messages or traces in Tivoli Log XML format.

**Installing the Tivoli XML Log Viewer**

The Tivoli XML Log Viewer is installed using an InstallShield MultiPlatform installation program. However, the installation program for the viewer is provided as a Java archive (JAR) file, and not as an executable file like the other product installation programs.

The installation JAR file is located on the IBM Tivoli Access Manager Base CD in the `/operating_system/xmllogviewer` directory. For example, on the IBM Tivoli Access Manager Base for Windows NT, Windows XP, Windows 2000 and Windows 2003 CD,
the location is \windows\xmllogviewer, and on the IBM Tivoli Access Manager Base for Solaris CD, the location is /solaris/xmllogviewer.

Because the InstallShield MultiPlatform installation program and the Tivoli XML Log Viewer are both Java applications, a JRE must be installed prior to installing and using the viewer. The same JRE used by Tivoli Access Manager can be used for the Tivoli XML Log Viewer. If a different JRE is used, the JRE must be at version 1.2.2 or later.

To install the viewer, perform the following steps:

1. Open a command prompt and navigate to the xmllogviewer directory on the installation media.
2. Enter the following command:
   
   java -cp setup.jar run

3. Navigate through the InstallShield MultiPlatform panels and select a location for the Tivoli XML Log Viewer, then proceed to install the viewer.
4. After the installation program completes, you can add the Tivoli XML Log Viewer directory to the command search path, if desired.
5. On UNIX systems, you might need to explicitly set execute permissions on the viewer.sh file:

   chmod +x viewer.sh

Running the Tivoli XML Log Viewer

The Tivoli XML Log Viewer is run by using the viewer script and specifying the name of one or more XML files along with the desired options. Output is directed to STDOUT in either HTML or text format. The output can be redirected to a file for viewing with a Web browser or text editor.

For example, to create an HTML file containing all of the messages from the policy and authorization servers sorted into chronological sequence, enter the following command:

   viewer msg__pdmgrd.xml msg__pdacld.xml > msg_19Oct2003_report.html

To display the messages from the policy server in text format, do the following:

   viewer -s text msg__pdmgrd.xml

Table 13 on page 41 shows sample text output from the Tivoli XML Log Viewer.

Additional information on the Tivoli XML Log Viewer, including how to tailor the output and how to uninstall it, can be found in the readme.html file, located in the same directory as the viewer script. This readme file also is provided in the xmllogviewer directory on the installation media.
Table 13. Sample text output from the Tivoli XML Log Viewer

2003-10-21-20:53:39.344 INFO HPDMS0467I Server startup mis/ivcore
2003-10-21-20:53:39.344 INFO HPDMS1778I Loading configuration mis/ivmgrd
2003-10-21-20:53:40.891 INFO HPDMS1785I Initialize client authorization mis/ivmgrd
2003-10-21-20:53:40.906 INFO HPDBA0243I Adding handler for command: (0x7), rc: (0x0). bas/mts
2003-10-21-20:53:40.906 INFO HPDBA0243I Adding handler for command: (0xe), rc: (0x0). bas/mts
2003-10-21-20:53:40.906 INFO HPDBA0243I Adding handler for command: (0xf), rc: (0x0). bas/mts
2003-10-21-20:53:40.906 INFO HPDBA0243I Adding handler for command: (0x6), rc: (0x0). bas/mts
2003-10-21-20:53:40.906 INFO HPDBA0243I Adding handler for command: (0x5), rc: (0x0). bas/mts
2003-10-21-20:53:40.906 INFO HPDBA0243I Adding handler for command: (0xa), rc: (0x0). bas/mts
2003-10-21-20:53:40.906 INFO HPDBA0243I Adding handler for command: (0xb), rc: (0x0). bas/mts
2003-10-21-20:53:40.906 INFO HPDBA0243I Adding handler for command: (0xf), rc: (0x0). bas/mts
2003-10-21-20:53:40.922 INFO HPDBA0243I Adding handler for command: (0xc), rc: (0x0). bas/mts
2003-10-21-20:53:40.922 INFO HPDBA0243I Adding handler for command: (0xd), rc: (0x0). bas/mts
2003-10-21-20:53:40.938 INFO HPDBA0243I Adding handler for command: (0x1), rc: (0x0). bas/mts
2003-10-21-20:53:40.938 INFO HPDBA0243I Adding handler for command: (0x2), rc: (0x0). bas/mts
2003-10-21-20:53:40.938 INFO HPDBA0243I Adding handler for command: (0x3), rc: (0x0). bas/mts
2003-10-21-20:53:40.938 INFO HPDBA0243I Adding handler for command: (0x8), rc: (0x0). bas/mts
2003-10-21-20:53:40.938 INFO HPDBA0243I Adding handler for command: (0x11), rc: (0x0). bas/mts
2003-10-21-20:53:40.938 INFO HPDBA0243I Adding handler for command: (0x105), rc: (0x0). bas/mts
2003-10-21-20:53:40.938 INFO HPDBA0243I Adding handler for command: (0x101), rc: (0x0). bas/mts
2003-10-21-20:53:40.953 INFO HPDBA0243I Adding handler for command: (0x302), rc: (0x0). bas/mts
2003-10-21-20:53:40.953 INFO HPDBA0243I Adding handler for command: (0x307), rc: (0x0). bas/mts
2003-10-21-20:53:40.953 INFO HPDBA0243I Adding handler for command: (0x103), rc: (0x0). bas/mts
2003-10-21-20:53:40.953 INFO HPDBA0243I Adding handler for command: (0x104), rc: (0x0). bas/mts
2003-10-21-20:53:40.953 INFO HPDBA0243I Adding handler for command: (0x305), rc: (0x0). bas/mts
2003-10-21-20:53:40.953 INFO HPDBA0243I Adding handler for command: (0x308), rc: (0x0). bas/mts
2003-10-21-20:53:40.953 INFO HPDBA0243I Adding handler for command: (0x205), rc: (0x0). bas/mts
2003-10-21-20:53:40.953 INFO HPDBA0243I Adding handler for command: (0x10), rc: (0x0). bas/mts
2003-10-21-20:53:40.953 INFO HPDBA0243I Adding handler for command: (0x306), rc: (0x0). bas/mts
2003-10-21-20:53:40.953 INFO HPDMS0467I Server ready mis/ivcore
2003-10-21-20:53:41.125 INFO HPDMS1785Iள The server is listening on port 7135. bas/mts
2003-10-21-20:59:32.547 ERROR HPDBB0753E The server is listening on port 7135. bas/mts
2003-10-21-20:59:32.563 ERROR HPDAC0458E The protected object name specified was not found in the authorization policy database. acl/acldb
2003-10-22-23:03:32.313 ERROR HPDMS1791I The protected object name specified was not found in the authorization policy database. acl/acldb
2003-10-22-23:03:32.313 ERROR HPDAC0458E Authentication failed. You have used an invalid user name, password or client certificate. ias/authsvc
2003-10-22-23:08:01.500 ERROR HPDC01364E The specified domain does not exist. icas/autsvc
2003-10-22-23:08:14.938 WARN HPDIA0219I An unknown user, zebra_admin, was presented to Tivoli Access Manager. icas/authsvc
Chapter 6. Trace logging

IBM Tivoli Access Manager (Tivoli Access Manager) provides configurable tracing capabilities that can aid in problem determination. Unlike message logging, trace logging (or tracing) is not enabled by default.

Trace data is intended primarily for use by IBM Customer Support for Tivoli software personnel and might be requested as part of diagnosing a reported problem. However, experienced product administrators can use trace data to diagnose and correct problems in the Tivoli Access Manager environment.

Tracing is best suited to situations where a problem is easily reproduced, is short-lived in duration, and can be produced without significant trace generation from other system activity. Note that enabling tracing can adversely affect the performance of the Tivoli Access Manager product and its associated products and applications.

Tracing can be activated when servers, daemons, and applications start by using routing files and Java properties files. In some cases, tracing can be activated dynamically using the pdadmin server task server_name trace set command (see special conditions in the section below).

Note: Tracing from the C language portions of the Tivoli Access Manager product is controlled through the use of routing files. Similarly, tracing from the Java language portions of Tivoli Access Manager is controlled through the use of Java properties files. Distinctions between the two methods of trace handling are mentioned when relevant.

This chapter covers the following topics:
- “Routing files” on page 44
- “Java properties files” on page 52
- “Mechanisms for controlling tracing”
- “Using the trace utility to capture actions” on page 48
- “Enabling tracing in Web Portal Manager” on page 54
- “Enabling tracing for WebSphere Application Server” on page 54
- “Enabling tracing for BEA WebLogic Server” on page 55

Mechanisms for controlling tracing

Tracing can be controlled using the following mechanisms:

routing file

A routing file can be used to control tracing of the Tivoli Access Manager policy server, authorization server, policy proxy server, WebSEAL server, and runtime component. The affected components must be stopped and restarted in order for modifications to the routing file to take effect.

Note: The Tivoli Access Manager Plug-in for Web Servers component sets the information that
Routing files are ASCII files that are used to customize the message logging and tracing associated with C language based Tivoli Access Manager servers, daemons, and other C language programs and applications. Information on tailoring message logging can be found in Chapter 5, “Tailoring message logging,” on page 27.

The contents of the routing file enables the user to control the following aspects of tracing:
- Whether tracing is enabled or disabled
- Where the trace output is to be directed
- If trace output is being directed to a file, how many files should be used, and how many trace entries should be placed in each file


### Location of routing files

The default locations for the routing files used by Tivoli Access Manager are shown in Table 14.

<table>
<thead>
<tr>
<th>Component</th>
<th>Default name and location of routing file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime environment</td>
<td>Windows %PD_HOME%etc\routing</td>
</tr>
<tr>
<td></td>
<td>Windows %PD_HOME%etc\routing</td>
</tr>
<tr>
<td></td>
<td>UNIX /opt/PolicyDirector/etc/routing</td>
</tr>
<tr>
<td>Policy server</td>
<td>Windows %PD_HOME%etc\pdmgrd_routing</td>
</tr>
<tr>
<td></td>
<td>Windows %PD_HOME%etc\pdmgrd_routing</td>
</tr>
<tr>
<td></td>
<td>UNIX /opt/PolicyDirector/etc/pdmgrd_routing</td>
</tr>
</tbody>
</table>
Table 14. Routing files associated with Tivoli Access Manager (continued)

<table>
<thead>
<tr>
<th>Component</th>
<th>Default name and location of routing file</th>
</tr>
</thead>
</table>
| Authorization server  | Windows %PD_HOME\%etc\pdacld_routing  
|                       | UNIX /opt/PolicyDirector/etc/pdacld_routing                                    |
| Policy proxy server   | Windows %PD_HOME\%etc\pdmgrproxyd_routing                                      |
|                       | UNIX /opt/PolicyDirector/etc/pdmgrproxyd_routing                               |
| WebSEAL server        | Windows %PD_WEB\%etc\routing                                                   |
|                       | UNIX /opt/pdweb/etc/routing                                                    |

Note: The Tivoli Access Manager Plug-in for Web Servers component sets the information that is typically contained within a routing file programmatically, therefore, it has no routing file of its own. The trace logging behavior of the Plug-in for Web Servers can be altered through the use of the $PD_SVC_ROUTING_FILE environment variable.

The default routing file can be overwritten through the use of the $PD_SVC_ROUTING_FILE environment variable. If the fully qualified file name indicated by $PD_SVC_ROUTING_FILE does not exist or is not accessible, the default routing file shown in Table 11 on page 28 is used.

Routing file trace entry

The format of a routing file entry that controls trace logging is:

```
comp:subcomp.level { [ ,subcomp.level] ... } :destination:location 
{ [ destination:location ]... } ;GOESTO:other_comp|severity
```

where:

- **comp**
  - Specifies the component to be traced.
  - An asterisk (*) can be used to specify all components.

- **subcomp**
  - Specifies a subcomponent to be traced.
  - An asterisk (*) can be used to specify all subcomponents.

- **level**
  - Reporting level. This is a number between 1 and 9 that specifies the maximum reporting level for trace entries. Level 1 specifies the least detailed output, only allowing level 1 trace entries. Level 9 specifies the most detailed output, allowing trace entries of any level (1–9).

- **destination**
  - Specifies where the trace output should be directed. Valid destinations are:
    - **STDOUT**
      - Trace entries are written to the standard output device as ASCII text in the current code page and locale.
XMLSTDOUT
Trace entries are written to the standard output device in Tivoli Log XML format.

STDERR
Trace entries are written to the standard error device as ASCII text in the current code page and locale.

XMLSTDERR
Trace entries are written to the standard error device in Tivoli Log XML format.

FILE
Trace entries are written to the specified file as ASCII text in the current code page and locale.

TEXTFILE
Same as FILE.

UTF8FILE
Trace entries are written to the specified file as UTF-8 text.

XMLFILE
Trace entries are written to the specified file in Tivoli Log XML format.

DISCARD
Trace entries are discarded.

If a destination of FILE, TEXTFILE, UTF8FILE, or XMLFILE is specified, it can optionally be followed by a period and two numbers separated by a period. These values indicate the number of files to be used and the number of message entries to be written to each file, respectively. If these values are omitted, one file is used and that file grows without limit.

location
Specifies the location for the trace log data.

When destination is STDOUT, STDERR, XMLSTDOUT, XMLSTDERR, or DISCARD, a hyphen (-) must be specified for location.

When destination is FILE, TEXTFILE, UTF8FILE, or XMLFILE, location is the fully qualified file name where the trace entries are written. The character string %ld can be used to insert the process ID into the file name.

When the number of files and the number of trace entries per file are specified as part of the destination, a period and the file number are added to the end of the file name specified.

On UNIX systems, the file name specification is followed by the desired file permissions, the user that owns the file, and the group that owns the file, each separated by a colon (:).

Note: On Windows systems, the file name must not end with a period when more than one file is used, because this results in a file name containing two consecutive periods, which is not valid.

GOESTO:other_comp | severity
Specifies that traces should additionally be routed to the same destinations and locations as either trace entries from the other_comp component, or messages of severity severity. This must be the last element in the trace logging specification.
Lines in the file that start with a pound sign (#) are considered comments and do not affect logging.

Routing file examples
To illustrate features available with the routing files, several examples are provided.

Trace logging in Tivoli XML Log format
To send trace output associated with the authorization server to a single file in Tivoli Log XML format, the %PD_HOME\etc\msg__pdacd.routing file could be modified as follows:
```
*:*.9:XMLFILE:E:\PDACLD_XMLTRACE\TRACE__PDACLD_%LD.XML
```

After stopping and restarting the authorization server to pick up the routing file change, and assuming that the process ID of the authorization server is 412, trace output is written to the following file:
```
E:\PDACLD_XMLTRACE\TRACE__PDACLD_412.XML
```

Use the Tivoli Log XML Viewer to analyze the XML file produced. See “Tivoli XML Log Viewer” on page 39 for instructions on installing and using the viewer.

Trace logging to multiple files
By default, trace logs grow without limit. This requires that the directories and file systems containing trace log files be checked on a periodic basis to ensure that enough space is available, and to prune the log files or make additional space available as necessary.

The routing files can be modified to limit the amount of disk space used for trace logging.

To send runtime tracing output, from reporting levels 1 through 5, to 10 different files, each containing a maximum of 10000 trace entries, the routing specification shown in Figure 14 could be used in the /opt/PolicyDirector/etc/routing file.
```
*:5:UTF8FILE.10.10000:/tmp/trace__am_utf8.log:666:ivmgr:ivmgr
```

Figure 14. Sending trace output to multiple files

Tracing starts with the first file, /tmp/trace__am_utf8.log.1. After 10000 trace entries have been logged to that file, trace entries are written to the second file, /tmp/trace__am_utf8.log.2. Tracing continues in this manner until 10000 trace entries are written to the /tmp/trace__am_utf8.log.10 file. At that point, the trace entries in the first file are deleted and tracing resumes again to the /tmp/trace__am_utf8.log.1 file.

Tracing a particular component
At the direction of IBM Customer Support for Tivoli software personnel, you might be asked to enable tracing for a particular component of Tivoli Access Manager. For example, if asked to trace the mgr component of the policy server, the /opt/PolicyDirector/etc/pdmgrd_routing file could be modified as shown in Figure 15 on page 48.
Determining maximum size of a trace log

Each entry made to a trace log file created using a destination of FILE, TEXTFILE, or UTF8FILE is an average of 200 bytes in size. The maximum size of a log file, in bytes, can be estimated as follows:

\[200 \times \text{(Number of log files)} \times \text{(Number of entries per log file)}\]

For example, given a specification of:

```
*:*.9:TEXTFILE.10.10000:C:/PROGRA~1/Tivoli/POLICY~1/log/trace__%ld.log
```

The maximum size would be approximately \((200 \times 10 \times 10000)\) or 20,000,000 bytes.

Trace entries written in Tivoli XML Log format are an average of 500 bytes in size, thus for a specification of:

```
*:*.9:XMLFILE.10.10000:/var/debug20031028A/trace__%ld.xml
```

The maximum size would be approximately \((500 \times 10 \times 10000)\) or 50,000,000 bytes.

Using the trace utility to capture actions

The trace utility allows you to capture information about error conditions and program control flow in Tivoli Access Manager Base, Tivoli Access Manager WebSEAL, and Tivoli Access Manager Plug-in for Web Servers. This information is stored in a file and used for debugging purposes. The trace utility is provided primarily to assist support personnel in diagnosing problems occurring with the functioning of the Tivoli Access Manager software.

As a user, you may find some of the Base, WebSEAL, and Plug-in for Web Servers tracing components useful. However, the majority are of little benefit unless you are diagnosing complex problems with the assistance of technical support personnel.

**Note**: Use trace with caution. It is intended as a tool to use under the direction of technical support personnel. Messages from trace are sometimes cryptic, are not translated, and can severely degrade system performance.

Basic trace command syntax

```
padmin> server task server-name trace command
```

The set of configured servers can be learned by issuing the following command:

```
padmin> server list
```

You can perform the following tasks with the `padmin trace` command:

- **trace list**
  - List all available trace components

- **trace set**
  - Enable the trace level and trace message destination for a component and its subordinates
trace show
    Show the name and level for all enabled trace components or for the
    specified component

List all available trace components
List the specified component or all components available to gather and report trace
information.
trace list [component]

Enable trace using pdadmin server task
Use the pdadmin server task server_name trace set command to enable the
gathering of trace information for the specified component and level.
trace set component level [file path=file | log-agent]

where:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component</td>
<td>The trace component name. This required argument indicates the component to be enabled. WebSEAL components are prefixed with pdweb.</td>
</tr>
<tr>
<td>level</td>
<td>Reporting level. This required argument must be in the range of 1 to 9. The level argument specifies the amount of detail gathered by the trace command. Level 1 specifies the least detailed output and level 9 specifies the most detailed output.</td>
</tr>
<tr>
<td>file</td>
<td>The fully qualified name of the file to which trace data will be written.</td>
</tr>
<tr>
<td>log-agent</td>
<td>Optionally specifies a destination for the trace information gathered for the specified component. See the event logging information in the IBM Tivoli Access Manager Base Administration Guide for details.</td>
</tr>
</tbody>
</table>

Show enabled trace components
List all enabled trace components or a specific enabled component. If a specified component is not enabled, no output is displayed.
trace show [component]

Example:
pdadmin> server task webseald-instance trace set pdweb.debug 2
pdadmin> server task webseald-instance trace show
pdweb.debug 2

Example WebSEAL trace output (pdweb.debug)
The pdweb.debug trace logs the HTTP headers associated with requests and responses. To log the message body as well, see “Example WebSEAL trace output (pdweb.snoop)” on page 50.

Note: The pdweb.debug component only operates at level 2.

The following command, entered as one line, invokes the trace utility for the pdweb.debug component at level 2, and directs the output to a file, using the event logging mechanism to specify a file log agent.
pdadmin> server task webseald-instance trace set pdweb.debug 2
    \ file path=/opt/pdweb/log/debug.log
Sample output of this command as it appears in the debug.log file (note the extensive text entries that follow the src_line value):

2003-08-11-23:42:19.725+00:00:001----- thread(7) trace.pdweb.debug:2
/amweb510/src/wand/wand/log.c:278: ----------- Browser ==> PD -----------
Thread_ID:27
GET /junction/footer.gif HTTP/1.1
Accept: */*
Referer: https://bevan/junction/
Accept-Language: en-us
Accept-Encoding: gzip, deflate
If-None-Match: "abe09-3c8-3b4cc0f2"
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.0.3705)
Host: bevan
Connection: Keep-Alive

2003-08-11-23:42:19.736+00:00:001----- thread(7) trace.pdweb.debug:2
/amweb510/src/wand/wand/log.c:278: ----------- PD ==> BackEnd -----------
Thread_ID:27
GET /footer.gif HTTP/1.1
via: HTTP/1.1 bevan:443
host: blade.cruz.ibm.com:444
user-agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.0.3705)
accept: */*
accept-language: en-us
accept-encoding: gzip, deflate
if-none-match: "abe09-3c8-3b4cc0f2"
referer: https://bevan/blade/
connection: close

2003-08-11-23:42:19.739+00:00:001----- thread(7) trace.pdweb.debug:2
/amweb510/src/wand/wand/log.c:278: ----------- PD <= BackEnd -----------
Thread_ID:27
HTTP/1.1 304 Not Modified
date: Wed, 11 Sep 2003 23:34:17 GMT
etag: "abe09-3c8-3b4cc0f2"
server: IBM_HTTP_SERVER/1.3.19.1 Apache/1.3.20 (Unix)
connection: close

Example WebSEAL trace output (pdweb.snoop)
The pdweb.snoop trace logs the HTTP headers as well as the message body associated with requests and responses. To trace only the message headers, see “Example WebSEAL trace output (pdweb.debug)” on page 49.

The following command, entered as one line, invokes the trace utility for the pdweb.snoop component at level 9, and directs the output to a file, using the event logging mechanism to specify a file log agent.

```
pdadmin> server task webseald-instance trace set pdweb.snoop 9 \ 
file path=/tmp/snoop.out
```
The sample output from this command shows a WebSEAL server sending 2137 bytes of data to a client at IP address 10.4.5.12:

```
2003-10-28-19:35:18.541+00:00:00:01---- thread(5) trace.pdweb.snoop.client:1
/home/amweb510/src/pdwebte/webcore/amw_snoop.cpp:159:
```

The trace subcomponent `pdweb.snoop.client` can be used to trace data sent between WebSEAL and clients. Use the `pdweb.snoop.jct` trace subcomponent to trace data sent between WebSEAL and junctions.

**Location and name of the trace log file**

Where any given trace log file is located and what it is named depends upon which Tivoli Access Manager component is being traced. For the Tivoli Access Manager authorization server, the trace log file can be explicitly specified by the user in the following command:

```
pdadmin> server task server-name trace set component level [file path=file]
```

where `server-name` is the name of the authorization server displayed by the `pdadmin server list` command, and `file` is the fully qualified trace file name.

Alternatively, if the Tivoli Access Manager Base routing file (discussed elsewhere within this document) is being used to enable and disable tracing, then the location and name of this trace log file can be defined within the routing file.

For WebSEAL, the trace log file can be explicitly specified by the user in the following command:

```
pdadmin> server task server-name trace set component level [file path=file]
```

where `server-name` is the name of the WebSEAL server displayed by the `pdadmin server list` command, and `file` is the fully qualified trace file name.

Alternatively, if the WebSEAL routing file (discussed elsewhere within this document) is being used to enable and disable tracing, then the location and name of this trace log file can be defined within the routing file.

For the Tivoli Access Manager Plug-in for Web Servers component, message log entries and trace log entries are always written, by default, to the same set of files when the Tivoli Common Directory is not configured. These include:
Windows:

(log for the authorization server)
`webpi-install-dir\log\msg__pdwebpi.log`

(log for the IIS plug-in)
`webpi-install-dir\log\msg__pdwebpi-iis.log`

UNIX:

(log for the authorization server)
`/var/pdwebpi/log/msg__pdwebpi.log`

(log for the watchdog server)
`/var/pdwebpi/log/msg__pdwebpimgr.log`

For the Tivoli Access Manager Plug-in for Edge Server component, message log entries and trace log entries are written to the same file. For this Tivoli Access Manager component, tracing is effectively enabled at all times. Tracing, therefore, is not controlled via the `pdadmin` command. This log file is:

Windows:
`edgepi-install-dir\cp\logs\event.date`

UNIX:
`/opt/ibm/edge/cp/server_root/logs/event.date`

Java properties files

Java properties files are ASCII files that are used to customize the message logging and tracing associated with Java language based Tivoli Access Manager servers, daemons, and other Java language programs and applications. Information on tailoring message logging can be found in Chapter 5, “Tailoring message logging,” on page 27.

The contents of the Java properties file enables the user to control the following aspects of tracing:

- Whether tracing is enabled or disabled
- Where the trace output is to be directed
- If trace output is being directed to a file, how many files should be used, and how big should each file be

Lines in the file that start with a pound sign (#) are comments and do not affect logging.

See Appendix B, “Sample Java application properties file,” on page 111, for a sample of a Java properties file.

Location of Java properties files

The default locations for the Java properties files used by Tivoli Access Manager are shown in Table 15 on page 53.
### Table 15. Java properties files associated with Tivoli Access Manager components

<table>
<thead>
<tr>
<th>Component</th>
<th>Default name and location of Java properties file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java application configured using the <code>com.tivoli.pdcfg.SvrSslCfg</code> command</td>
<td>The output application configuration file as specified in the <code>com.tivoli.pdcfg.SvrSslCfg</code> command.</td>
</tr>
<tr>
<td>Java language based Tivoli Access Manager commands, such as <code>pdjrtecfg</code> and <code>com.tivoli.pdcfg.SvrSslCfg</code> or applications not explicitly configured.</td>
<td><code>$JAVA_HOME/PolicyDirector/ PDJLog.properties</code></td>
</tr>
</tbody>
</table>
| Web Portal Manager | Windows $PD_HOME%\java\export\pdwpm\pdwpm.properties  
UNIX `/opt/PolicyDirector/java/export/pdwpm/pdwpm.properties` |
| WebSphere Application Server integration support | `$PDWAS_HOME/etc/jlog.properties` |
| BEA WebLogic Server integration support | `$JAVA_HOME/amwls/wls_domain_name/wls_realm_name/amwlsjlog.properties` |

**Note:** If the `com.tivoli.pdcfg.SvrSslCfg` command has not been run, then no application-specific configuration file exists and the `PDJLog.properties` file is used.

### Trace loggers

Each properties file contains properties for one or more trace loggers. The `isLogging` property specifies whether trace logging is enabled or not. To turn tracing on for a specific trace logger, use:

```java
baseGroup.trace_logger_name.isLogging=true
```

To disable logging for a specific trace logger, use:

```java
baseGroup.PDJapp_nameTraceLogger.isLogging=false
```

Associated with each trace logger is one or more file handlers. A file handler specifies the destination for a specific class, or severity, of messages. Once trace logging is enabled by the trace logger, the file handler properties are examined to determine if traces should be logged and, if so, how and where. The properties associated with a file handler are:

```java
baseGroup.PDJapp_nameTraceFileHandler.fileName=
baseGroup.PDJapp_nameTraceFileHandler.maxFileSize=
baseGroup.PDJapp_nameTraceFileHandler.maxFiles=
```

where:

- **fileName** Specifies the fully qualified file name to be used as the base name for trace log files. The file can be in any location accessible by the Java application.

- **maxFileSize** Specifies the maximum size, in KB, of each trace log file. Default is 512.

- **maxFiles** Specifies the maximum number of files to be used for trace logging. Default is 3.
Enabling trace logging for the Java runtime environment

Trace logging for components using the Tivoli Access Manager runtime environment is controlled through the application-specific properties file or, for applications not explicitly configured with the `com.tivoli.pd.jcfg.SvrSslCfg` command, the `$JAVA_HOME/PolicyDirector/PDJLog.properties` file. To enable tracing:

```
baseGroup.PDJapp_nameTraceLogger.isLogging=true
```

For each trace logger, the properties file defines a mask attribute, `baseGroup.PDJapp_nameTraceAllMaskFilter.mask`, that determines what levels of tracing are enabled. Valid mask values are 1 through 9. The precise meaning of any given mask value is unimportant. The general intention is that ascending from a lower mask value to a higher mask value (1 to 2, for example) increases the level of information detail that is traced.

Setting the mask value to a particular level means that all tracing levels up to and including the specified level are enabled. For example, if the mask value is 4, then tracing levels 1, 2, 3, and 4 are traced.

Enabling tracing in Web Portal Manager

**Note:** Only change entries in this properties file at the request of IBM Customer Support for Tivoli software.

The properties file associated with Web Portal Manager is `pdwpm.properties`.

**Windows**  
%PD_HOME%\java\export\pdwpm\pdwpm.properties

**UNIX**  
/opt/PolicyDirector/java/export/pdwpm/pdwpm.properties

Trace logging is disabled by default.

```
baseGroup.PDJamwpm-host_nameTraceLogger.isLogging=false
```

Enabling tracing for WebSphere Application Server

The properties file associated with Tivoli Access Manager WebSphere Application Server integration support is `$PDWAS_HOME/etc/jlog.properties`. To enable tracing for all trace loggers:

```
baseGroup.TraceLogger.isLogging=true
```

Trace logging for other trace loggers can be selectively enabled using:

```
baseGroup.trace_logger.isLogging=true
```

and disabled using:

```
baseGroup.trace_logger.isLogging=false
```

If the `isLogging` property is not set for a specific trace logger, the setting of `baseGroup.TraceLogger.isLogging` is used.

The available trace loggers are:

```
message_logger  Description
AMWASWebTraceLogger  Enables tracing for the WebSphere Application
```
Server authorization plug-in. To diagnose most problems, usually only this trace logger needs to be enabled.

**AmasRBPFTraceLogger** Enables tracing for the Role Based Policy Framework. This is the underlying framework used by the WebSphere Application Server integration support to make access decisions.

**AmasCacheTraceLogger** Enables tracing for the policy caches used by the Role Based Policy Framework.

After changing the properties file, enable the change in the WebSphere Application Server console:

**Version 5**
1. Click **Servers → Application Servers** in the left frame.
2. Click on the desired server.
3. Click **Logging and Tracing → Diagnostic Trace**.
4. Under the Trace Specifications heading, click **Modify**.
5. Click **Components → PDWAS** and select the desired trace level.
6. Click **Apply**.

**Version 4**
1. Click **Servers → Application Servers** in the left frame.
2. Click on the desired server.
3. Click **Logging and Tracing Properties → Diagnostic Trace Service**.
4. In the **Trace Specification** box, enter the following:
   ```
   com.ibm.ws.security.PDWAS=all=enabled
   ```
5. Click **Apply**.

The **Trace Specification** should now indicate that tracing is enabled at the desired level. Save the configuration change, and then restart the server for this change to take effect.

As mentioned in [Chapter 4, “Default message logging,” on page 19](#), messages and traces are written to the **SystemOut.log** file associated with the WebSphere Application Server.

### Enabling tracing for BEA WebLogic Server

The properties file associated with Tivoli Access Manager BEA WebLogic Server integration support is:

```
$JAVA_HOME/amwls/wls_domain_name/wls_realm_name/amwlsjlog.properties
```

where **wls_domain_name** is the name of your configured BEA WebLogic Server domain, and **wls_realm_name** is the name of your configured BEA WebLogic Server realm within the domain. Trace logging is disabled by default. To enable tracing for all trace loggers:

```
baseGroup.TraceLogger.isLogging=true
```

Tracing for other trace loggers can be selectively enabled using:

```
baseGroup.trace_logger.isLogging=true
```
and disabled using:

```java
baseGroup.trace_logger.isLogging=false
```

If the `isLogging` property is not set for a specific trace logger, the setting of `baseGroup.TraceLogger.isLogging` is used.

The available trace loggers are:

<table>
<thead>
<tr>
<th>trace_logger</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AmasRBPFTraceLogger</td>
<td>Enables tracing for the Role Based Policy Framework. This is the underlying framework used by the BEA WebLogic Server integration support to make access decisions.</td>
</tr>
<tr>
<td>AmasCacheTraceLogger</td>
<td>Enables tracing for the policy caches used by the Role Based Policy Framework.</td>
</tr>
<tr>
<td>AMSSPICfgTraceLogger</td>
<td>Enables tracing for the configuration components of the BEA WebLogic Server integration support.</td>
</tr>
<tr>
<td>AMSSPIAuthzTraceLogger</td>
<td>Enables tracing for the authorization components of the BEA WebLogic Server integration support.</td>
</tr>
<tr>
<td>AMSSPIAuthnTraceLogger</td>
<td>Enables tracing for the authentication components of the BEA WebLogic Server integration support.</td>
</tr>
<tr>
<td>AMSSPIRoleMapperTraceLogger</td>
<td>Enables tracing for programmatic role checks.</td>
</tr>
<tr>
<td>AMSSPIResourceManagerTraceLogger</td>
<td>Enables tracing associated with the interfaces between BEA WebLogic Server and the Role Based Policy Framework.</td>
</tr>
</tbody>
</table>

After changing the properties file, restart the BEA WebLogic Server to pick up the change.

As mentioned in [Chapter 4, “Default message logging,” on page 19](#), messages and traces are written to the log file associated with the BEA WebLogic Server.

---

**Tailoring tracing in WebSphere Edge Server**

Tivoli Access Manager Plug-in for Edge Server does not make use of a routing file. The trace log settings are defined within the object space definition configuration file, `osdef.conf`. The default location for this file is:

- **Windows** `edgepi_install_dir\etc`
- **UNIX** `/var/ibm/edge/etc`

Relevant sections of the `osdef.conf` file are:

```ini
# Indicates the category of logging messages that the plug-in will send to the event log file. Only messages matching the specified category will be sent to the log file. This option is valid only in the [Global] section of this file. The default value is 'EWI'.
```

Log message flag options:
```
# 'E' - Error messages
# 'W' - Warning messages
# 'I' - Informational messages
# 'D' - Debugging messages
```

---

56 IBM Tivoli Access Manager for e-business: Problem Determination Guide
logging_flags = EWI

# Indicates the verbosity level for informational and debugging log
# messages. This value can range from 0 to 5. A higher number means
# that more messages will be logged. This option is valid only in the
# [Global] section of this file. The default value is '3'.
#logging_level = 3

## Trace log entry format

Figure 16 shows an example of a trace entry taken from a Tivoli Access Manager
trace log file. The trace entry shown has been split into multiple lines for
readability.

2003-10-29-18:01:06.984-06:00I—> pdmgrd DEBUG8 mgr general
e:\am510\src\vmgrd\pdmgrapi\MrMgmtDomainMan.cpp 736 0x000007d0
CII ENTRY: MrMgmtDomainMan::setCurrentDomainName

Figure 16. Sample trace log entry in text format

This trace entry consists of the following fields:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
<th>Value in sample entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>The time that the trace entry was made.</td>
<td>2003-10-29-18:01:06.984-06:00I</td>
</tr>
<tr>
<td></td>
<td>This time is in the following format:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CCYY-MM-DD–hh:mm:ss.fff[+</td>
<td>–]ii:jjI, where:</td>
</tr>
<tr>
<td></td>
<td>CCYY-MM-DD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Year, month, and day</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hh:mm:ss.fff</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hours, minutes, seconds, and fractional seconds</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ii:jjI Time inaccuracy factor</td>
<td></td>
</tr>
<tr>
<td>process_name</td>
<td>Name of the process which created the entry</td>
<td>pdmgrd</td>
</tr>
<tr>
<td>reporting_level</td>
<td>Reporting level of the trace entry.</td>
<td>DEBUG8</td>
</tr>
<tr>
<td>component</td>
<td>Component associated with process that generated entry.</td>
<td>mgr</td>
</tr>
<tr>
<td>subcomponent</td>
<td>Subcomponent associated with process that generated entry.</td>
<td>general</td>
</tr>
<tr>
<td>src_file</td>
<td>Name of the product source file that generated the entry</td>
<td>e:\am510\src\vmgrd\pdmgrapi\MrMgmtDomainMan.cpp</td>
</tr>
<tr>
<td>src_line</td>
<td>Line number in source file</td>
<td>736</td>
</tr>
<tr>
<td>thread_id</td>
<td>Thread ID, in hexadecimal</td>
<td>0x000007d0</td>
</tr>
<tr>
<td>text</td>
<td>Text of the trace entry</td>
<td>CII ENTRY: MrMgmtDomainMan::setCurrentDomainName</td>
</tr>
</tbody>
</table>

## Sample trace output

The Tivoli Access Manager trace log example shown in Table 17 on page 58 was
generated by adding the following line to the pdmgrd_routing file:

*:*.9:UTF8FILE:C:/TRACELOG/20031029A_UTF8.LOG
Table 17. Sample trace output from the policy server

<table>
<thead>
<tr>
<th>Time</th>
<th>Process</th>
<th>Function</th>
<th>File/Line</th>
<th>Args</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003-10-29-18:03:46.109-06:00</td>
<td>pdmgrd</td>
<td>DEBUG8 ira_get_dn_utf8()</td>
<td>e:\am510\src\ivrgy\ira_auth.c</td>
<td>847 0x00000b80</td>
<td>CII ENTRY: ira_get_dn_utf8() parm: irmgrd/master</td>
</tr>
<tr>
<td>2003-10-29-18:03:46.109-06:00</td>
<td>pdmgrd</td>
<td>DEBUG8 ira_suffixes_get()</td>
<td>e:\am510\src\ivrgy\ira_auth.c</td>
<td>171 0x00000b80</td>
<td>CII ENTRY: ira_suffixes_get() parm: Default</td>
</tr>
<tr>
<td>2003-10-29-18:03:46.109-06:00</td>
<td>pdmgrd</td>
<td>DEBUG8 ira_determine_ldap_server_type()</td>
<td>e:\am510\src\ivrgy\ira_auth.c</td>
<td>1867 0x00000b80</td>
<td>CII ENTRY: ira_determine_ldap_server_type()</td>
</tr>
<tr>
<td>2003-10-29-18:03:46.109-06:00</td>
<td>pdmgrd</td>
<td>DEBUG7 ldap_init()</td>
<td>e:\am510\src\ivrgy\ira_handle.c</td>
<td>632 0x00000b80</td>
<td>ldap_init() server: ale: port: 389</td>
</tr>
<tr>
<td>2003-10-29-18:03:46.109-06:00</td>
<td>pdmgrd</td>
<td>DEBUG7 ldap_simple_bind_s()</td>
<td>e:\am510\src\ivrgy\ira_handle.c</td>
<td>654 0x00000b80</td>
<td>ldap_simple_bind_s() DN: cn=ivmgrd/master,cn=SecurityDaemons,secAuthority=Default</td>
</tr>
<tr>
<td>2003-10-29-18:03:46.109-06:00</td>
<td>pdmgrd</td>
<td>DEBUG7 ldap_simple_bind_s(): No timeout - calling ldap_simple_bind_s</td>
<td>e:\am510\src\ivrgy\ira_ldap.c</td>
<td>1240 0x00000b80</td>
<td>ldap_simple_bind_s(): No timeout - calling ldap_simple_bind_s</td>
</tr>
<tr>
<td>2003-10-29-18:03:46.141-06:00</td>
<td>pdmgrd</td>
<td>DEBUG7 ldap_search_ext_s()</td>
<td>e:\am510\src\ivrgy\ira_search_ex_c</td>
<td>2645 0x00000b80</td>
<td>ldap_search_ext_s() base: scope: 0 filter: (objectclass=)</td>
</tr>
</tbody>
</table>
Chapter 7. Using AutoTrace

AutoTrace is a multi-platform set of debug trace and analysis tools designed to reduce the time required to identify and resolve software problems. The AutoTrace tool is to be used under the direction of an IBM field support representative. The majority of the information within this chapter is of interest only to IBM field support personnel. Customer-oriented directions for reporting AutoTrace information to IBM Customer Support are contained in the section “Using AutoTrace to collect trace data” on page 66.

Section topics:
- “AutoTrace overview” on page 59
- “Building the Tivoli Access Manager product with AutoTrace” on page 60
- “Description of AutoTrace files” on page 61
- “Verifying that AutoTrace is functioning properly” on page 65
- “Using AutoTrace to collect trace data” on page 66
- “AutoTrace output format and example” on page 67
- “Useful AutoTrace commands” on page 69
- “How to create a trace ID file” on page 71

AutoTrace overview

AutoTrace is a multi-platform set of debug trace and analysis tools designed to reduce the time required to identify and resolve software problems. AutoTrace has not only been identified as a tool that can improve the serviceability of the IBM Tivoli Access Manager product, but also seems likely to become a common debug trace tool across the set of Tivoli software products. Version 3.1.5 of AutoTrace is incorporated into this release of Tivoli Access Manager.

Typically the AutoTrace tool is to be used under the direction of an IBM field support representative.

Features of AutoTrace 3.1.5 include:
- AutoTrace 3.1.5 automatically adds trace instrumentation during a product’s build. At a minimum, instrumented code includes function entry and exit points, but may also include code blocks and thrown and caught exceptions. AutoTrace also traces the values of simple data types that are passed into and returned from each instrumented function.

Debug trace instrumentation beyond what AutoTrace provides automatically can also be added to product source code manually by developers using the AutoTrace API.

- AutoTrace becomes an integral part of the product’s build environment. It does not modify product source code to add its debug trace instrumentation. Instead, it adds its instrumentation to the preprocessor output of the compiler which is then compiled. All instrumented executables are then linked with an AutoTrace shared library.

- During the product’s build, AutoTrace assigns a unique trace ID (integer) to each instrumented function (one trace ID is associated with both the function entry and the function exit). A trace ID is also assigned to each trace hook added to the code manually using the AutoTrace API.
A file containing a list of these trace ID’s can be used to enable/disable tracing for the associated trace points. Individual trace ID’s can also be enabled/disabled to fine-tune the set of functions being traced.

How a file of trace ID’s can be created will be discussed later.

- AutoTrace records trace data within a memory segment that is shared between the process performing the tracing and the AutoTrace `atctl.exe` process which owns/manages the memory segment.

  The AutoTrace `config` file is used to define one or more shared memory segments—also known as tracing "channels". The `config` file is also used to configure which products or which processes trace to any given channel.

  AutoTrace provides facilities for copying a snapshot of the trace data within a channel to a file. This can be accomplished either via an AutoTrace command or from product code via the AutoTrace API. The contents of this file can then be viewed using other AutoTrace tools.

- AutoTrace includes powerful trace analysis and trace profiling utilities which allow the trace data snapshots to be analyzed either locally or remotely. These include:
  - A text report generator which generates a text report from the captured trace snapshot
  - An HTML reporter which generates an HTML report from the trace snapshot and enables it to be viewed either locally or remotely
  - A snapshot file viewer (the most powerful of the trace analysis tools)

Each of these tools reports configurable detail about a trace entry including:

- Time stamp
- Process ID
- Thread ID
- Program name
- Source file name and line number
- Product ID

AutoTrace works alongside the Tivoli Access Manager debug trace facilities which were introduced in previous releases. It does not replace them.

---

**Building the Tivoli Access Manager product with AutoTrace**

AutoTrace version 3.1.5 has been integrated with the Tivoli Access Manager Base and WebSEAL components on the Microsoft Windows, IBM AIX, and Sun Solaris Operating Environment platforms. AutoTrace automatically adds debug trace instrumentation to all C and C++ code. All executables and shared libraries built as part of the Tivoli Access Manager Base and WebSEAL are linked with an AutoTrace shared library named `libatrc`. This library is installed along with the Tivoli Access Manager runtime.

As AutoTrace instruments each function within the build, it adds to a database file which contains information about each instrumented function. This information includes the function name, the assigned trace ID, the name of the file containing the function, and an associated function signature computed from several items including the name of the function, the names and types of the function’s parameters, and the function’s return type. This database is used by other AutoTrace tools that do not participate in the build process. This file can be
thought of as a database of information which associates a given product function with a specific trace ID. A trace ID is an integer used to enable and disable tracing for the associated function.

AutoTrace also supports an exclude file which contains a list of functions which are not to be automatically instrumented by AutoTrace. These might include functions which are invoked frequently and/or functions which are known to be reliable. The AutoTrace exclude file is mentioned here only to alert the reader that not all functions are necessarily instrumented by AutoTrace.

### Description of AutoTrace files

The following AutoTrace files are installed as part of the Tivoli Access Manager runtime:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>atserv.exe (Windows only)</td>
<td>A Windows service required by AutoTrace. Unless this service is installed and started, the AutoTrace control program (atctl(.exe)) will fail to initialize. This service is installed and started when the Tivoli Access Manager Runtime is installed.</td>
</tr>
<tr>
<td>libatrc.dll (Windows) libatrc.a (AIX) libatrc.so (Solaris) libatrc.sl (HP-UX)</td>
<td>The AutoTrace shared library with which each product executable and shared library is linked.</td>
</tr>
<tr>
<td>atctl(.exe)</td>
<td>The AutoTrace control program. When the atctl(.exe) process initializes, it reads information from two AutoTrace configuration files named config and product. It uses this information to allocate the AutoTrace shared-memory segment(s) where trace data is gathered. The atctl program will be initialized automatically each time the system is booted. atctl(.exe) commands are also invoked by the user to control tracing.</td>
</tr>
<tr>
<td>atinstall.exe (Windows only)</td>
<td>Enables the automatic installation and starting of atserv.exe.</td>
</tr>
<tr>
<td>atprintf(.exe)</td>
<td>A tool that can be used to manually add comments or additional trace information to a trace channel.</td>
</tr>
<tr>
<td>atrpt(.exe)</td>
<td>Used for viewing and analyzing the contents of a trace snapshot file. The AutoTrace database is used by atrpt(.exe) and can be specified via either: - the -d command line option - the ATRC_DB environment variable - atrpt(.exe) also allows the user to specify the location of the database file after atrpt(.exe) has been started.</td>
</tr>
</tbody>
</table>

| AccessManagerBaseAutoTraceDatabaseFile.obfuscated AccessManagerWebSEALAutoTraceDatabaseFile.obfuscated | Obfuscated copies of the AutoTrace database files which mask Tivoli Access Manager file and function names. |
Table 18. AutoTrace file descriptions (continued)

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>config file</td>
<td>The config file contains information which configures the number of tracing channels and the size of each, as well as which products and/or processes which will trace to each. The directory which contains this file is communicated to AutoTrace on the atctl init &lt;dir-name&gt; command each time the machine is rebooted. Up to 255 tracing channels can be defined. This file along with the product file establishes the configuration of the AutoTrace control channel and tracing channels. Typically each product will ship its own config file. Tivoli Access Manager Base and WebSEAL each ship their own config files.</td>
</tr>
<tr>
<td>product file</td>
<td>The product file contains an entry which contains the product’s assigned 8-digit ID and the product short name. The directory which contains this file must be communicated to AutoTrace using the atctl init &lt;dir-name&gt; command each time the machine is rebooted. This file along with the config file establishes the configuration of the AutoTrace control channel and tracing channels. Typically each product will ship its own product file. Tivoli Access Manager Base and WebSEAL each ship their own product files.</td>
</tr>
</tbody>
</table>

The following AutoTrace files are not included with Tivoli Access Manager:

Table 19. AutoTrace files not included with Tivoli Access Manager

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>atdb(.exe)</td>
<td>Used for manipulating and extracting information from the AutoTrace database. The location of the AutoTrace database can be specified to atdb(.exe) either with the -d command line option, or via the ATRC_DB environment variable.</td>
</tr>
<tr>
<td>atprof(.exe)</td>
<td>The AutoTrace trace profiler. This tool enables the user to determine which functions in the instrumented product are the most heavily used. This information can be used to tune the set of trace ID’s contained within either a “first failure data capture” trace ID file or an AutoTrace exclude file.</td>
</tr>
<tr>
<td>AccessManagerBaseAutoTraceDatabaseFile</td>
<td></td>
</tr>
<tr>
<td>AccessManagerWebSEALAutoTraceDatabaseFile</td>
<td></td>
</tr>
<tr>
<td>AMWebPIAutoTraceDatabaseFile</td>
<td>The master AutoTrace database files created by the respective product builds. They contain information about each instrumented function which includes the function name, the assigned trace ID, the name of the file containing the function, and an associated function signature computed from several items including the name of the function, the names and types of the function’s parameters and the function’s return type. A function’s signature is computed during the build to enable AutoTrace to determine whether a trace ID already exists for this function within the database.</td>
</tr>
</tbody>
</table>
Description of the AutoTrace product file

The product file contains an entry which associates a product’s IBM assigned 8-digit ID with a product identifier. The assigned 8-digit ID is a component of every AutoTrace trace hook added to that product. The product identifier is a 3 character short name used on various AutoTrace commands to identify trace hooks or trace IDs for that particular product or component. The product identifier for Tivoli Access Manager Base is HPD and the identifier for WebSEAL is DPW. See “Product identifiers” on page 2 for a complete list of product identifiers in Tivoli Access Manager.

The directory which contains the product file must be communicated to AutoTrace using the atctl init dir-name command each time the machine is rebooted. Tivoli Access Manager does this automatically.

This file, along with the config file, establishes the configuration of the AutoTrace tracing channels. Tivoli Access Manager Base and WebSEAL each ship their own product files.

The AutoTrace product file for Tivoli Access Manager Base contains the following information:

<table>
<thead>
<tr>
<th>Product</th>
<th>Name</th>
<th>Home</th>
</tr>
</thead>
<tbody>
<tr>
<td>49420005</td>
<td>HPD</td>
<td>-</td>
</tr>
</tbody>
</table>

The AutoTrace product file for Tivoli Access Manager WebSEAL contains the following information:

<table>
<thead>
<tr>
<th>Product</th>
<th>Name</th>
<th>Home</th>
</tr>
</thead>
<tbody>
<tr>
<td>49420006</td>
<td>DPW</td>
<td>-</td>
</tr>
</tbody>
</table>

The product file for each of the products above can be found in that product’s etc directory.

Description of the AutoTrace config file

The config file contains information which configures the number of tracing channels and the size of each, as well as the products and/or processes which will trace to each. The directory which contains this file must be communicated to AutoTrace on the atctl init dir-name command each time the machine is rebooted. Up to 255 tracing channels can be defined.

This file along with the product file establishes the configuration of the AutoTrace tracing channels.

Tivoli Access Manager Base and WebSEAL each ship its own AutoTrace config file.

The AutoTrace config file for Tivoli Access Manager Base contains the following information which defines:

- A default size for each AutoTrace tracing channel
- Trace channel 37 for the pdmgrd process
- Trace channel 38 for the pdacld process
- Trace channel 39 for Tivoli Access Manager Base.

Refer to the config file itself for more details regarding the nature of its contents.
The AutoTrace config file for Tivoli Access Manager WebSEAL contains the following information:

```
process pdmgrd:
  chan 37

process pdacld:
  chan 38

product HPO:
  chan 39
```

The config file for each of the products above can be found in that product’s etc directory.

Configuration information within a product’s AutoTrace config file can be used to direct all of that product’s trace output to a single trace channel. For example:

```
product ABC:
  chan 5
```

It is also possible to direct the trace output of a specific process of that product to its own trace channel. For example:

```
process myABCprocess:
  chan 6
```

When channels for both a product and a process within a given product are specified to AutoTrace, the process stanza will take precedence over the product stanza. In other words, trace data generated by the process will be directed to the trace channel associated with that process. Trace data generated by other processes within that product will be directed to the trace channel associated with the product.

Assume that AutoTrace has been configured to trace two different products. Product DEF and product HIJ are configured as follows within their respective AutoTrace config files:

```
product DEF:
  chan 7
process myDEFprocess:
  chan 8

product HIJ:
  chan 9
```

Let’s also assume that myDEFprocess calls functions which reside within a shared library of product HIJ. If trace ID’s are enabled within myDEFprocess and within product HIJ’s shared library, then the trace information for both will be recorded within trace channel 8.
Description of the AutoTrace exclude file

The AutoTrace exclude file is used only during the product’s build. It is not shipped with the product.

The AutoTrace exclude file contains a list of functionname filename pairs that identify functions which are not to be automatically instrumented by AutoTrace during the product’s build. Functions that are candidates for inclusion within the exclude file include heavily used functions and those functions which are generally regarded as stable.

The AutoTrace exclude file is described here only to serve as a reminder that not every product function has necessarily been instrumented by AutoTrace. The contents of the exclude file has no effect on any trace hooks that have been manually added to the code via the AutoTrace API. No manual AutoTrace hooks have been used by Tivoli Access Manager in this release.

Verifying that AutoTrace is functioning properly

Initializing atserv.exe (Windows only)

On the Windows platform only, AutoTrace 3.1.5 provides a Windows service that manages the shared memory segment containing the tracing channels. The executable name is atserv.exe. This service should be installed and started automatically by the installation of the Tivoli Access Manager runtime.

From the Windows Control Panel, click on the Services icon and check whether atserv.exe was successfully installed and started. It will be listed as the “AutoTrace Runtime”. If not, locate the following file:
C:\Program Files\Tivoli\Policy Director\sbin\atinstall.exe

Execute the following command:
MSDOS> atinstall.exe --quietinstall "c:\Program Files\Tivoli\Policy Director\sbin"

This command should cause atserv.exe to be installed as a service and to be started. Observe that --quietinstall option uses two dashes.

Initializing atctl.(exe)

The AutoTrace control program (atctl.exe) is located in the .../Policy Director/sbin directory. This program should automatically be initialized after the Tivoli Access Manager Base product is installed, and again after each system boot.

Verify that the AutoTrace control program has been properly initialized by entering the following commands.

The atctl info command should produce output similar to the following when atctl.exe has initialized successfully:
C:\Program Files\Tivoli\Policy Director\sbin> atctl info
CHAN SIZE VER NLIFE
0 1M 3 0
35 1M 3 0
37 1M 3 0
38 1M 3 0

Chapter 7. Using AutoTrace 65
A channel defined to AutoTrace using the `atctl init` command may not appear within the `atctl info` output until AutoTrace has actually attempted to trace to that channel.

The `atctl config` command should produce output similar to the following when Tivoli Access Manager Base and WebSEAL are properly installed and configured:

```
C:\Program Files\Tivoli\Policy Director\sbin> atctl config
# last modified: Wed May 29 16:04:06 2002
default:
    size 1M
    llen 16
product 49420005:  # HPD
    chan 39
product 49420006:  # DPW
    chan 36
process webseald:
    chan 35
process pdmgrd:
    chan 37
process pdacld:
    chan 38
```

The `atctl product` command should produce output similar to the following when Tivoli Access Manager Base and WebSEAL are properly installed and configured:

```
C:\Program Files\Tivoli\Policy Director\sbin> atctl product
# prod name home
  49420005 HPD C:\PROGRA~1\Tivoli\POLICY\1\etc
  49420006 DPW C:\PROGRA~1\Tivoli\PDWeb\etc
```

If any of the above commands fail, `cd` to the `Policy Director\sbin` directory where `atctl.exe` is located and enter the following commands:

1. (to reinitialize AutoTrace for Tivoli Access Manager Base)
   ```
   atctl init ..\etc
   ```

2. (to reinitialize AutoTrace for Tivoli Access Manager WebSEAL)
   ```
   atctl init ..\..\PDWeb\etc
   ```

   where the paths specified are the paths to that product’s config and product files.

   The `atctl info`, `atctl config`, and `atctl product` commands above should now all execute successfully.

**Using AutoTrace to collect trace data**

1. Create a file of trace ID’s which are appropriate for the problem being debugged. These trace ID’s are used to enable and disable debug tracing within the associated functions.

   Techniques for how to create a trace ID file are described later in this chapter. Typically a file of trace ID’s will be provided to the customer by a representative from IBM field support.

2. The selected trace ID’s can be enabled using the following command:

   ```
   atctl on product_identifier traceID_filename
   ```

   See "Product identifiers" on page 2 for the list of identifiers in Tivoli Access Manager. These can also be obtained from each product’s associated AutoTrace product file.
Test whether these trace ID’s were successfully enabled using the following command:
```
atctl trace product_identifier
```

A list of the trace IDs which have been enabled should appear. These should match the trace IDs contained within the trace ID file that was used to enable them.

3. Run the failing product scenario to generate trace data.

4. The following command can be used to capture a snapshot of the trace data within the desired tracing channel to a file:
```
atctl snap channel-num
```

The trace data will be copied to a snap file within the current directory with the name `snap%c_%u.at` where `%c` is the channel number, and `%u` is an automatically assigned unique number between 00 and 99. When a channel’s trace contents are “snapped”, all trace data within that channel is cleared.

5. A text version of the snap file can be generated using the following command:
```
atrpt -t snapfile-spec > textfile
```

Alternatively, a snap file viewer can be started using the command `atrpt` with no arguments. This command presents the user with a GUI that provides the user with a means for selecting the snap file to be viewed and the associated AutoTrace database file.

All AutoTrace commands that need to know the name of the AutoTrace database file look to find that information within an environment variable named ATRC_DB. If this environment variable has not been defined to identify the AutoTrace database file, the name of the AutoTrace database file must be explicitly specified using the `-d` command line option.

IBM customers should stop at this point and deliver the AutoTrace snap file to qualified IBM customer support personnel for interpretation.

---

**AutoTrace output format and example**

The following is an example of an AutoTrace trace entry as it would appear if the snapshot of the trace channel were formatted such that all available fields were included in the formatted output. This can be accomplished using the following command:
```
atrpt -t -O* -d AccessManagerBaseAutoTraceDatabaseFile full-snapfile-path
```

where `-O*` specifies that all available fields be included in the output.

```
11279 49420005 19605 pdmgrd 3136 0 15:30:28.578386 |
ira_ldap_search_ext_s(ld=0x1d7b6b8, base=0x1bdcd2f8, scope=2, filter=0x1d713e0, 
attrs=0x12f2f78, attrsonly=0, serverctrls=(nil), clientctrls=(nil), 
timeout=0x12f140, sizelimit=2048, res=0x12f14c) [ira_ldap.c,502]
```

Which AutoTrace database file, such as Base or WebSEAL, is specified on the `atrpt` command is dependent on the snap file being formatted. For example, if the snap file was captured from a WebSEAL trace channel, then the WebSEAL AutoTrace database file must be used to format it.
This AutoTrace trace entry consists of the following fields:

Table 20. AutoTrace entry descriptions

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>line_number</td>
<td>The line number of this entry within the trace. ('&quot;11279&quot; in the example above)</td>
</tr>
<tr>
<td>product_id</td>
<td>The AutoTrace product ID assigned to the product which generated the trace entry. ('&quot;49420005&quot; in the example above)</td>
</tr>
<tr>
<td>trace_id</td>
<td>The AutoTrace trace ID associated with the trace hook which generated the trace entry. ('&quot;19605&quot; in the example above)</td>
</tr>
<tr>
<td>process_name</td>
<td>The name of the process which generated the trace entry. ('&quot;pdmgrd&quot; in the example above)</td>
</tr>
<tr>
<td>process_id</td>
<td>The process ID of the process which generated the trace entry. ('&quot;3136&quot; in the example above)</td>
</tr>
<tr>
<td>thread_id</td>
<td>The thread ID of the thread which generated the trace entry. ('&quot;0&quot; in the example above)</td>
</tr>
<tr>
<td>absolute_time</td>
<td>The actual time at which the trace entry was captured. ('&quot;15:30:28.578386&quot; in the example above)</td>
</tr>
<tr>
<td>text</td>
<td>The text of the trace entry. ('&quot;ira_ldap_search_ext_s(ld=0x1d7b6b8, base=0x1bdcdf8, scope=2, filter=0x1d713e0, attrs=0x12f278, attrsonly=0, serverctrls=(nil), clientctrls=(nil), timeoutp=0x12f140, sizelimit=2048, res=0x12f14c)&quot; in the example above.)</td>
</tr>
<tr>
<td>src_file</td>
<td>The name of the product source file which generated the entry. ('&quot;ira_ldap.c&quot; in the example above)</td>
</tr>
<tr>
<td>src_line</td>
<td>The line number within the product source file where the entry was generated. ('&quot;502&quot; in the example above)</td>
</tr>
</tbody>
</table>

Below is a portion of an actual AutoTrace trace file. All of the formatting performed by the AutoTrace report generator, atrpt, has been lost in the output presented below.

```
11278  49420005  16335  pdmgrd  3136  0  15:30:28.578383 |
ira_internal_search(ld=0x1d7b6b8, base=0x1bdcdf8, scope=2, filter=0x1d713e0, attrs=0x12f278, sizelimit=2048, entries=0x12f298) [ira_entry.c,2349]

11279  49420005  19605  pdmgrd  3136  0  15:30:28.578386 |
ira_ldap_search_ext_s(ld=0x1d7b6b8, base=0x1bdcdf8, scope=2, filter=0x1d713e0, attrs=0x12f278, attrsonly=0, serverctrls=(nil), clientctrls=(nil), timeoutp=0x12f140, sizelimit=2048, res=0x12f14c) [ira_ldap.c,502]

11280  49420005  19605  pdmgrd  3136  0  15:30:28.578388 |
ira_ldap_search_ext_s: line 636 IRA_ldap.c [ira_ldap.c,636]

11281  49420005  19605  pdmgrd  3136  0  15:30:28.585992 |
ira_ldap_search_ext_s: line 636  IRA_ldap.c [ira_ldap.c,636]

11282  49420005  16335  pdmgrd  3136  0  15:30:28.585996 |
ira_internal_search: line 2378  IRA_entry.c [ira_entry.c,2378]

11283  49420005  16335  pdmgrd  3136  0  15:30:28.585999 |
ira_internal_search: line 2381  IRA_entry.c [ira_entry.c,2381]

11284  49420005  16335  pdmgrd  3136  0  15:30:28.586003 |
ira_internal_search: line 2391  IRA_entry.c [ira_entry.c,2391]
```
It is not expected that customers will be performing AutoTrace traces except under the direction of IBM Customer Support personnel. Therefore, customers will typically be sending their AutoTrace snap files to IBM Customer Support for analysis, rather than attempting to format them themselves.

### Useful AutoTrace commands

**Table 21. AutoTrace commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>atctl remove all</code></td>
<td>Clear any configuration or trace data that may exist within shared memory. If this command is run, <code>atctl init</code> must be re-run.</td>
</tr>
<tr>
<td><code>atctl init dir-name</code></td>
<td>Initialize AutoTrace using the config and product files within the directory <code>dir-name</code>. AutoTrace must be initialized in this fashion once per system reboot. This command initializes the AutoTrace shared memory (control channel 0 and the tracing channel(s)). Typically, each product is responsible for issuing its own <code>atctl init</code> command specifying its own copy of config and product.</td>
</tr>
<tr>
<td><code>atctl info</code></td>
<td>Examine the channel information to determine whether <code>atctl init</code> was successful.</td>
</tr>
</tbody>
</table>
| `atctl product` | Display the following information for each product initialized:  
  - product ID  
  - product name  
  - home directory  
  `atctl` obtains “product ID” and “product name” from the product file. |
| `atctl config` | Display the current configuration (that is, the information within control channel 0). |
Table 21. AutoTrace commands (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>atctl config config-file</code></td>
<td>Process the config file and store its information into control channel 0.</td>
</tr>
<tr>
<td><code>atctl version</code></td>
<td>Display the AutoTrace version.</td>
</tr>
<tr>
<td><code>atctl on HPD all</code></td>
<td>Activate all trace IDs associated with the HPD product.</td>
</tr>
<tr>
<td><code>atctl off HPD all</code></td>
<td>Deactivate all trace IDs associated with the HPD product.</td>
</tr>
<tr>
<td><code>atctl on HPD &lt;traceID-file&gt;</code></td>
<td>Activate all trace IDs associated with the HPD product that are specified within the file traceID-file. The traceID-file can be a list of integers and/or integer ranges (14025-14050 represents an integer range).</td>
</tr>
<tr>
<td><code>atctl off HPD traceID-file</code></td>
<td>Deactivate all trace IDs associated with the HPD product that are specified within the file traceID-file.</td>
</tr>
<tr>
<td><code>atctl trace HPD</code></td>
<td>List all trace IDs which are active for product HPD.</td>
</tr>
<tr>
<td><code>atctl snap channel-num</code></td>
<td>Capture a snapshot of the trace data within the specified channel. The data will be copied to a &quot;snap file&quot; within the current directory with the name snap%c_%u.at where %c is the channel number, and %u is an automatically assigned unique number between 00 and 99. The snap file can then be examined with atrpt. When a channel’s trace contents are &quot;snapped&quot;, all trace data within that channel is cleared.</td>
</tr>
<tr>
<td><code>atctl snap channel-num file-spec</code></td>
<td>Capture a snapshot of the trace data within the specified channel. Place the trace data in the specified snap file. The snap file can then be examined with atrpt. When a channel’s trace contents are “snapped”, all trace data within that channel is cleared.</td>
</tr>
<tr>
<td><code>atctl suspend channel-num</code></td>
<td>Suspend tracing to a given channel.</td>
</tr>
<tr>
<td><code>atctl resume channel-num</code></td>
<td>Resume tracing to a given channel.</td>
</tr>
<tr>
<td><code>atctl reset channel-num</code></td>
<td>Resets the given channel, thus erasing all trace data that it contains.</td>
</tr>
<tr>
<td><code>atrpt -ti snapfile-spec</code></td>
<td>Display information about the captured snapshot (date/time/machine/channel).</td>
</tr>
<tr>
<td><code>atrpt -t snapfile-spec &gt; textfile</code></td>
<td>Create a text copy of the captured trace snapshot.</td>
</tr>
</tbody>
</table>
Table 21. AutoTrace commands (continued)

| NOTE: Some trace fields, such as thread id, are not, by default, included within the atrpt output. |
| To add the thread id: atrpt -t -0 tid snapfile-spec > textfile |
| To add the filename and line number where each trace entry was generated: atrpt -t -0 filename -0 line snapfile-spec > textfile |

| atrpt snapfile-spec |
| Display the contents of a snapfile using the snap file viewer. |

| REMINDER: The atrpt command requires the complete file pathname of the AutoTrace database file generated by the build. This file may be specified either via the -d <file-spec> argument on the atrpt command line, or it may be specified via the ATRC_DB environment variable. |
| atdb get id |
| If id is a function name, return its associated trace ID. If id is a trace ID, return its associated function name. |

| UNIX: atdb get * |
| Windows: atdb get * |
| Get all of the trace IDs in the AutoTrace database file along with all of the associated function and file names. This is a good way to begin creating a trace ID file. |

| atdb group inputfile(s) > ffdcfile |
| Read a file of function name/file name pairs, and generate a file which contains the associated trace IDs from the AutoTrace database file. A function name may be specified either as simple function name, or as a simple function name and the associated file name separated by one or more spaces. The wildcard characters “*” and “?” may also be used within function and file names. |

| This is how a trace ID file would typically be created for collective activation/deactivation at runtime. |
| More than one input file can be specified: atdb group inputfile1 inputfile2 > ffdcfile |

| REMINDER: The atdb command requires the complete file pathname of the AutoTrace database file generated by the build. This file may be specified either via the -d <file-spec> argument on the atdb command line, or it may be specified via the ATRC_DB environment variable. |
| atdb is not shipped with the Tivoli Access Manager product. It is intended to be used only by IBM field support personnel. |

How to create a trace ID file

The information within this section is intended for IBM field support personnel.
An AutoTrace trace ID is a unique integer value associated with a function within the compiled product code. Information about each trace ID is stored within the AutoTrace database file created during the product build process. To view all of these trace IDs along with their associated functions, enter the following command:

```
atdb -d database-name get id *
```

where `database-name` represents the name of the AutoTrace database for the product of interest.

Separate AutoTrace databases exist for the Tivoli Access Manager Base and WebSEAL. The name of the AutoTrace database file for the Tivoli Access Manager Base is `AccessManagerBaseAutoTraceDatabaseFile`. An obfuscated version of this file which masks function and file names is shipped with the product. It cannot be used to create a trace ID file.

A file of trace IDs is used for collectively activating or deactivating the associated trace points. Two techniques are described below for creating a trace ID file. Both use AutoTrace tools for extracting trace IDs from the AutoTrace database file.

**Technique 1**

Use the following command to get all of the trace IDs from the AutoTrace database along with all associated function and file names and copy this information into the file `traceIDfile`:

```
atdb -d database-name get id * > traceIDfile
```

This file can then be manually edited to include only the set of desired trace IDs.

Use the following command to activate tracing for the trace IDs defined within the `traceIDfile` (where HPD represents the short name of the Tivoli Access Manager Base product):

```
atctl on HPD traceIDfile
```

Use a different short name if activating trace points for another product.

**Technique 2**

Create a file of "function-name and file-name" pairs for the functions you want to trace. In this discussion, this file is called `functionlistfile`. Use the `atdb` command to extract the trace IDs associated with those functions from the AutoTrace database file as follows:

```
atdb -d database-name group functionlistfile > traceIDfile
```

The wildcard characters "*" and "?" can be used within either the "function-name" or the "file-name". For the Windows platform, when specifying a specific filename, use the .c or .cpp file name extension, as you would expect. On UNIX platforms, use a .i file name extension instead (for example, `myfilename.i`).

Use the following command to activate tracing for the trace IDs defined within the `traceIDfile` (where HPD represents the short name of the Tivoli Access Manager Base product).

```
atctl on HPD traceIDfile
```

Use a different short name if activating trace points for another product.

Technique 2 is superior to technique 1.
**Known problems**

AutoTrace allocates each tracing channel separately from shared memory. The Solaris platform has an environment variable which, by default, limits allocations of shared memory to a maximum of 1024k bytes.

Modifying the AutoTrace config file to specify a channel size greater than 1024k will cause the allocation of the AutoTrace shared memory segment to fail unless the Solaris environment variable has been updated to allow shared memory allocations which are larger. This is accomplished by adding the following line to the /etc/system file and rebooting:

```
set shmsys:shminfo_shmmax=0xffffffff
```

If the allocation of the AutoTrace shared memory segment should fail, the performance of the associated Tivoli Access Manager product will be noticeably slower. AutoTrace reports its failed shared memory allocation attempt within the system error log.
Chapter 8. Using WebSEAL statistics

IBM Tivoli Access Manager WebSEAL provides a series of built-in software modules that, when enabled, can monitor specific server activity and collect information about those activities. At any instance in time, you can display the statistics information gathered since that module was enabled. In addition, you can direct this statistics information to log files.

When you display statistics information, you see a "snapshot" of the information since the module was enabled. The information gathered by WebSEAL statistics provides a relative view of the activity being recorded. If statistics are captured at regular intervals over a period of time, you can generate a graphical view of the relative relationship of the server activities.

pdadmin stats command syntax

Use the pdadmin stats command to manage statistics components. This section describes the valid operations for the pdadmin stats command:

Basic pdadmin stats command

pdadmin> server task webseal-instance_name stats command

You can perform the following tasks with the pdadmin stats command:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stats on</td>
<td>Enable statistics dynamically, by component</td>
</tr>
<tr>
<td>stats off</td>
<td>Disable statistics, by component or all components at once</td>
</tr>
<tr>
<td>stats show</td>
<td>List enabled components</td>
</tr>
<tr>
<td>stats get</td>
<td>Display current statistics values by component or all components at once</td>
</tr>
<tr>
<td>stats reset</td>
<td>Reset statistics values by component or all components at once</td>
</tr>
<tr>
<td>stats list</td>
<td>List all available statistics components</td>
</tr>
</tbody>
</table>

Enable statistics dynamically

You can enable statistics reporting dynamically with the pdadmin stats on command or statically with configuration parameters in the WebSEAL configuration file.

Use the pdadmin stats on command to enable statistics gathering and set the statistics report frequency, count, and destination for a component.

stats on component [interval [count]] [logagent]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>component</td>
<td>The statistic component name. Required. Statistics are gathered in WebSEAL memory for this component. Statistics for this component can also be recorded in a log file by specifying the optional arguments to this command.</td>
</tr>
</tbody>
</table>
Table 23. pdadmin stats command arguments (continued)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interval</td>
<td>The time interval between reports of information. This argument is optional and results in statistics being sent to a log file. When this option is specified, statistics are sent, by default, to standard out of the WebSEAL server, which is the WebSEAL log file. You can specify another output location using the logagent argument. If interval is not specified, no statistics are sent to any log file. However, the statistic component is still enabled. You can obtain reports dynamically at any time using pdadmin stats get.</td>
</tr>
<tr>
<td>count</td>
<td>The number of reports sent to a log file. This argument is optional and requires that the interval argument be specified. If interval is specified without count, the duration of reporting is indefinite. After the count value is reached, reporting to a log file stops. However, the statistic component is still enabled. You can obtain reports dynamically at any time using pdadmin stats get.</td>
</tr>
<tr>
<td>logagent</td>
<td>Optionally specifies a destination for the statistics information gathered for the specified component. Refer to the “Using event logging” chapter of the IBM Tivoli Access Manager Base Administration Guide for complete configuration details.</td>
</tr>
</tbody>
</table>

Note: By default, the pdweb.http, pdweb.doccache, and pdweb.jmt components are always enabled and cannot be disabled.

See also “Enabling statistics statically using event logging” on page 83

Example 1:

This example enables the pdweb.http component. Because the interval option is not specified, you can only obtain statistics information for this component dynamically using pdadmin stats get.

    pdadmin> server task webseal-instance stats on pdweb.http

Example 2:

This example enables the pdweb.http component. Because the interval argument is specified, the information is sent (by default) to the standard WebSEAL log file. The interval and count arguments cause the log file to accumulate 100 entries representing statistics reports 20 seconds apart.

    pdadmin> server task webseal-instance stats on pdweb.http 20 100

Example 3:

This example enables the pdweb.http component. The logagent argument uses event logging configuration to specify a destination file for the statistics information. The interval argument (without a count value) sends statistics information for this component to the log file every 20 seconds indefinitely. The growth of the log file is controlled by the rollover_size parameter. Refer to the "Using event logging" chapter of the IBM Tivoli Access Manager Base Administration Guide for complete event logging configuration details.

    pdadmin> server task webseal-instance stats on pdweb.http 20 file
    path=/tmp/jmt-stats.log,rollover_size=1,flush_interval=20

Example 4:
This example illustrates a limitation of managing statistics dynamically. The first command enables the `pdweb.http` component and directs statistics information to the A.log file. The second command attempts to activate a second log file, B.log. However, this action actually results in deactivating A.log while activating B.log.

```
pdadmin> server task webseald-instance stats on pdweb.http 20 file=/tmp/A.log
pdadmin> server task webseald-instance stats on pdweb.http 20 file=/tmp/B.log
```

**Disable statistics**

Disable statistics gathering for a component or all components at once.

```
stats off [component]
```

**Example:**

```
pdadmin> server task webseald-instance stats off pdweb.sescache
```

**Note:** By default, the `pdweb.threads`, `pdweb.doccache`, and `pdweb.jmt` components are always enabled and cannot be disabled.

**Show enabled statistics components**

List all enabled statistics components or a specific enabled component. If a specified component is not enabled, no output is displayed.

```
stats show [component]
```

**Example 1:**

```
pdadmin> server task webseald-instance stats show
pdweb.authn
pdweb.doccache
pdweb.jmt
pdweb.sescache
pdweb.threads
```

**Example 2:**

```
pdadmin> server task webseald-instance stats show pdweb.authn
pdweb.authn
```

**Get current statistic values dynamically**

Show the current values of statistics being gathered by a component or all enabled components.

```
stats get [component]
```

**Example:**

```
pdadmin> server task webseald-instance stats get pdweb.threads
active: 4
total: 50
```

**Reset statistics values**

Reset the values being gathered by an enabled component or all enabled components at once.

```
stats reset [component]
```

**Example:**

```
pdadmin> server task webseald-instance stats reset pdweb.threads
```
List all available statistics components

List all components available to gather and report statistics.
stats list

Example:
pdadmin> server task webseald-instance stats list
pd.ras.stats.monitor
pd.log.EventPool.queue
pd.log.file.clf
pd.log.file.ref
pd.log.file.agent
pdweb.authn
pdweb.authz
pdweb.http
pdweb.https
pdweb.threads
pdweb.jmt
pdweb.sescache
pdweb.doccache
pdweb.jct.1

Statistics components and activity types

This section describes the statistics components available to Tivoli Access Manager WebSEAL.

pdweb.authn component

The pdweb.authn statistics component gathers information related to WebSEAL authentication. The following table describes the types of information available:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pass</td>
<td>The total number of successful authentications</td>
</tr>
<tr>
<td>fail</td>
<td>The total number of failed authentications</td>
</tr>
<tr>
<td>pwd exp</td>
<td>The total number of authentication attempts made with an expired password</td>
</tr>
<tr>
<td>max</td>
<td>The maximum time for a single authentication process</td>
</tr>
<tr>
<td>avg</td>
<td>The average time for a single authentication process</td>
</tr>
<tr>
<td>total</td>
<td>The total time for all authentication processing</td>
</tr>
</tbody>
</table>

Example:
pdadmin> server task webseald-instance stats get pdweb.authn
pass : 2
fail : 1
pwd exp : 0
max : 0.178
avg : 0.029
total : 0.382

pdweb.authz component

The pdweb.authz statistics component gathers information related to WebSEAL authorization. The following table describes the types of information available:
Table 25. pdweb.authz information types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pass</td>
<td>The total number of successful authorization requests (how many resources were successfully accessed)</td>
</tr>
<tr>
<td>fail</td>
<td>The total number of failed authorization requests</td>
</tr>
</tbody>
</table>

Example:
```
pdadmin> server task webseald-instance stats get pdweb.authz
pass : 2
fail : 1
```

**pdweb.http component**

The `pdweb.http` statistics component gathers information related to WebSEAL HTTP communication. The following table describes the types of information available:

Table 26. pdweb.http information types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reqs</td>
<td>The total number of HTTP requests received</td>
</tr>
<tr>
<td>max-worker</td>
<td>The maximum time used by a single worker thread to process an HTTP request</td>
</tr>
<tr>
<td>total-worker</td>
<td>The total time used by all worker threads that process HTTP requests</td>
</tr>
<tr>
<td>max-webseal</td>
<td>The maximum time used to process a single HTTP request - measured inside the worker thread, after the request headers have been read, and eliminating connection setup overhead</td>
</tr>
<tr>
<td>total-webseal</td>
<td>The total time used to process all HTTP requests - measured inside the worker threads, after the request headers have been read, and eliminating connection setup overhead</td>
</tr>
</tbody>
</table>

Example:
```
pdadmin> server task webseald-instance stats get pdweb.http
reqs : 0
max-worker : 0.000
total-worker : 0.000
max-webseal : 0.000
total-webseal : 0.000
```

**pdweb.https component**

The `pdweb.https` statistics component gathers information related to WebSEAL HTTPS communication. The following table describes the types of information available:

Table 27. pdweb.https information types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reqs</td>
<td>The total number of HTTPS requests received</td>
</tr>
<tr>
<td>max-worker</td>
<td>The maximum time used by a single worker thread to process an HTTPS request</td>
</tr>
<tr>
<td>total-worker</td>
<td>The total time used by all worker threads that process HTTPS requests</td>
</tr>
<tr>
<td>max-webseal</td>
<td>The maximum time used to process a single HTTPS request - measured inside the worker thread, after the request headers have been read, and eliminating connection setup overhead</td>
</tr>
</tbody>
</table>

Chapter 8. Using WebSEAL statistics
Table 27. pdweb.https information types (continued)

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>total-webseal</td>
<td>The total time used to process all HTTPS requests - measured inside the</td>
</tr>
<tr>
<td></td>
<td>worker threads, after the request headers have been read, and eliminating</td>
</tr>
<tr>
<td></td>
<td>connection setup overhead</td>
</tr>
</tbody>
</table>

Example:
```
pdadmin> server task webseal-instance stats get pdweb.https
reqs : 0
max-worker : 0.000
total-worker : 0.000
max-webseal : 0.000
total-webseal : 0.000
```

**pdweb.threads component**

The **pdweb.threads** statistics component gathers information related to WebSEAL worker thread activity. This component is always enabled by default and cannot be disabled. The following table describes the types of information available:

Table 28. pdweb.threads information types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active</td>
<td>The total number active worker threads handling requests</td>
</tr>
<tr>
<td>total</td>
<td>The total number of configured worker threads</td>
</tr>
</tbody>
</table>

Example:
```
pdadmin> server task webseal-instance stats get pdweb.threads
active : 0
total : 50
```

**pdweb.jmt component**

The **pdweb.jmt** statistics component gathers information related to the WebSEAL junction mapping table. This component is always enabled by default and cannot be disabled. The following table describes the types of information available:

Table 29. pdweb.jmt information types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hits</td>
<td>The total number of requests that required URL mapping via the junction</td>
</tr>
<tr>
<td></td>
<td>mapping table</td>
</tr>
</tbody>
</table>

Example:
```
pdadmin> server task webseal-instance stats get pdweb.jmt
hits : 5
```

**pdweb.sescache component**

The **pdweb.sescache** statistics component gathers information related to the WebSEAL session/credential cache activity. The following table describes the types of information available:
Table 30. pdweb.sescache information types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hit</td>
<td>The number of requests that resulted in a session cache hit - that is, the</td>
</tr>
<tr>
<td></td>
<td>user had a session cache entry and it was successfully referenced</td>
</tr>
<tr>
<td>miss</td>
<td>The number of requests that missed a session cache hit</td>
</tr>
<tr>
<td>add</td>
<td>The number of entries that have been added to the session cache</td>
</tr>
<tr>
<td>del</td>
<td>The number of entries that have been deleted from the cache</td>
</tr>
<tr>
<td>inactive</td>
<td>The number of entries removed from the cache because the inactivity</td>
</tr>
<tr>
<td></td>
<td>timeout value had expired</td>
</tr>
<tr>
<td>lifetime</td>
<td>The number of entries removed from the cache because the lifetime</td>
</tr>
<tr>
<td></td>
<td>timeout value had expired</td>
</tr>
<tr>
<td>LRU expired</td>
<td>The number of times a &quot;Least Recently Used&quot; cache entry is expired or</td>
</tr>
<tr>
<td></td>
<td>removed to make room for a new entry.</td>
</tr>
</tbody>
</table>

Example:
```
pdadmin> server task webseald-instance stats get pdweb.sescache
hit : 0
miss : 0
add : 0
del : 0
inactive : 0
lifetime : 0
LRU expired : 0
```

**pdweb.doccache component**

The **pdweb.doccache** statistics component gathers information related to WebSEAL document caching activity. This component reports statistics for all MIME types enabled in the [content-cache] stanza of the WebSEAL configuration file. This component is always enabled by default and cannot be disabled.

The following table describes the types of global information available for all MIME types:

Table 31. pdweb.doccache information types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Errors</td>
<td>The number of errors reported by the pdweb.doccache component when the there are memory allocation failures, initialization failures, and invalid MIME type header values</td>
</tr>
<tr>
<td>Uncachable</td>
<td>The number of instances when there is no cache defined for the MIME type of the document to be cached</td>
</tr>
<tr>
<td>Pending Deletes</td>
<td>The number of entries marked for deletion, but are still in use</td>
</tr>
<tr>
<td>Pending Size</td>
<td>The number of bytes used by entries marked for deletion but are still in use</td>
</tr>
<tr>
<td>Misses</td>
<td>The number of times a URL is looked up in the document cache and not found. A found cached document eliminates the need to access the real document again.</td>
</tr>
<tr>
<td>Cache MIME type</td>
<td>The MIME type of documents stored in this cache.</td>
</tr>
</tbody>
</table>
Cache MIME type statistics:

Table 32. Cache MIME type statistics

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max size</td>
<td>The maximum combined byte size of all documents in the cache.</td>
</tr>
<tr>
<td>Max entry size</td>
<td>The maximum byte size for any single cached document. If the document size exceeds this internally calculated value, it is not cached.</td>
</tr>
<tr>
<td>Size</td>
<td>The total byte count for all documents currently residing in the cache.</td>
</tr>
<tr>
<td>Count</td>
<td>The current number of entries in the cache.</td>
</tr>
<tr>
<td>Hits</td>
<td>The number of successful lookups (documents successfully found in the cache)</td>
</tr>
<tr>
<td>Stale hits</td>
<td>The number of successful lookups that found an entry that was too old and was purged instead</td>
</tr>
<tr>
<td>Create waits</td>
<td>The number of times subsequent requests for a document are blocked (made to wait) while the document content is initially being cached</td>
</tr>
<tr>
<td>Cache no room</td>
<td>The number of times a document that is valid for caching cannot fit into the cache because there are too many entries being created at the same time</td>
</tr>
<tr>
<td>Additions</td>
<td>The number of successful new entries in the cache</td>
</tr>
<tr>
<td>Aborts</td>
<td>The number of times the creation of a new cache entry is aborted because of problems or a header that indicates the entry should not be cached</td>
</tr>
<tr>
<td>Deletes</td>
<td>The number of cache entries deleted because the entry is stale (expired) or the creation was aborted</td>
</tr>
<tr>
<td>Updates</td>
<td>The number of entries that have had expiry times updated</td>
</tr>
<tr>
<td>Too big errors</td>
<td>The number of attempts to cache documents that exceed the maximum entry size (and therefore are not cached)</td>
</tr>
<tr>
<td>MT errors</td>
<td>The number of times more than one thread tries to create the same entry in the cache (MT=Multi-Threading)</td>
</tr>
</tbody>
</table>

Example:

```plaintext
pdadmin> server task webseald-instance stats get pdweb.doccache
General Errors : 0
Uncachable : 0
Pending Deletes: 0
Pending Size : 0
Misses : 0
Cache MIME type : text/html
  Max size : 2048000
  Max entry size : 128000
  Size : 0
  Count : 0
  Hits : 0
  Stale hits : 0
  Create waits : 0
  Cache no room : 0
  Additions : 0
  Aborts : 0
  Deletes : 0
  Updates : 0
  Too big errors : 0
  MT errors : 0
```
pdweb.jct.# component

The pdweb.jct.# statistics component gathers information related to WebSEAL junctions. The following table describes the types of information available:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>The actual junction name (listed as a number in the command)</td>
</tr>
<tr>
<td>reqs</td>
<td>The total number of requests routed across this junction</td>
</tr>
<tr>
<td>max</td>
<td>The maximum time consumed in a single request across this junction</td>
</tr>
<tr>
<td>total</td>
<td>The total time consumed by requests across this junction</td>
</tr>
</tbody>
</table>

Example:

```
padmin> server task webseald-instance stats get pdweb.jct.1
[/]
reqs : 0
max  : 0.000
total: 0.000
```

Enabling statistics statically using event logging

Statistics can be statically configured in the [aznapi-configuration] stanza of the WebSEAL configuration file by using the stats parameter to enable the statistics interface and using the logcfg event logging parameter to direct the statistics information to a destination:

```
[aznapi-configuration]
stats = component [interval [count]]
logcfg = stats.component:destination
```

Refer to "Enable statistics dynamically" on page 75 for information on the interval and count arguments.

Refer to the "Using event logging" chapter of the IBM Tivoli Access Manager Base Administration Guide for complete event logging configuration details.

Example 1:

In this example, the stats parameter enables the component and any interval and count arguments. The logcfg parameter specifies the destination for this information. Refer to the "Using event logging" chapter of the IBM Tivoli Access Manager Base Administration Guide for complete configuration details.

```
[aznapi-configuration]
stats = pdweb.jmt 20
logcfg = stats.pdweb.jmt:file path=/tmp/jmt.log,rollover_size=-1,flush=20
```

Example 2:

In this example, multiple stats parameters enable multiple components. Multiple logcfg parameters specify multiple destinations. Note that, unlike the dynamic stats on command, you can specify several destination files for the same component. Refer to the "Using event logging" chapter of the IBM Tivoli Access Manager Base Administration Guide for complete event logging configuration details.

```
[aznapi-configuration]
stats = pdweb.jmt 20
stats = pdweb.authn 40
```
stats = pdweb.jct.1 50
logcfg = stats.pdweb.jmt:file path=/tmp/jmtA.log,rollover_size=-1,flush=20
logcfg = stats.pdweb.jmt:file path=/tmp/jmtB.log,rollover_size=-1,flush=20
logcfg = stats.pdweb.authn:file path=/tmp/an.log,rollover_size=-1,flush=20
logcfg = stats.pdweb.jct.1:file path=/tmp/jct.log,rollover_size=-1,flush=20
Chapter 9. Basic troubleshooting

Section topics:
- “Verifying software installation” on page 85
- “Verifying the functionality of individual components” on page 86

Verifying software installation

The installation of Tivoli Access Manager involves the installation and configuration of a number of prerequisite software products and Tivoli Access Manager components. Operational failures for the Tivoli Access Manager product can result from the failure to install all required prerequisite software, or the failure to install the correct level of this software.

Always re-examine your installed software and software levels if either:
- A new Tivoli Access Manager installation fails to work properly
- An existing Tivoli Access Manager installation fails to work properly after updating one or more prerequisite products or Tivoli Access Manager components

The Tivoli Access Manager installation guides and Release Notes provide detailed information about specific software requirements that must be satisfied before Tivoli Access Manager can be successfully installed and configured. These requirements include supported operating systems, prerequisite software, and required patches. Be sure to note the release or level of each software item.

The Tivoli Access Manager technical documentation also describes exactly what software must be installed on any given Tivoli Access Manager system. For example, the IBM Tivoli Access Manager Base Installation Guide informs you that a system containing the Tivoli Access Manager policy server must, at a minimum, have the following software installed:
- IBM Global Security Kit (GSKit)
- IBM Tivoli Directory Client
- Tivoli Access Manager runtime environment
- Tivoli Access Manager policy server

Tools exist that enable you to determine whether the correct software, and level, is installed upon any given Tivoli Access Manager system. Tivoli Access Manager supplies a utility named pdversion that automatically lists all Tivoli Access Manager components installed on that system along with their version numbers.

The pdversion utility does not list any prerequisite software, such as the IBM Tivoli Directory Client or the IBM Global Security Kit (GSKit). In most cases, the presence or absence of this prerequisite software can be determined using native operating system utilities.
- For the Windows platforms, use the Add/Remove Programs icon on the Windows Control Panel
- For AIX, use the lslpp –l command
- For Solaris, use the pkginfo –l command
- For Linux, use the rpm –qa command
• For HP-UX, use the `swlist` command

Even with these tools, certain packages on certain platforms might still be difficult to locate. GSKit on the Windows platforms, for example, adds no entry to the Windows Add/Remove Program list. On the Windows platform, GSKit is typically installed in the following directory:

```
C:\Program Files\IBM\gsk7
```

The following command can be used on Windows systems to display the GSKit version:

```
C:\Program Files\IBM\gsk7\bin\gskver
```

Use these tools in conjunction with the software requirements listed within the Tivoli Access Manager installation guides to ensure that all the required software is installed upon each machine within your Tivoli Access Manager configuration.

---

**Verifying the functionality of individual components**

A typical installation of the Tivoli Access Manager product involves a number of prerequisite products and Tivoli Access Manager components. Tivoli Access Manager will experience operational problems if any of these prerequisite products are not installed and configured properly.

This section describes how a user can verify whether certain prerequisite software products or Tivoli Access Manager components are generally operating properly. If any operational irregularities are discovered, refer to the appropriate installation and configuration instructions for that product or Tivoli Access Manager component.

The following table illustrates the components that are required with various types of Tivoli Access Manager systems.

<table>
<thead>
<tr>
<th>Tivoli Access Manager System</th>
<th>Required Software Components</th>
</tr>
</thead>
</table>
| Policy server               | - IBM Global Security Kit (GSKit)  
                              | - Tivoli Access Manager runtime  
                              | - IBM Tivoli Directory Client  
                              | - Tivoli Access Manager policy server |
| Runtime system              | - IBM Global Security Kit (GSKit)  
                              | - Tivoli Access Manager runtime  
                              | - IBM Tivoli Directory Client  |
| Java runtime system         | - IBM Global Security Kit (GSKit)  
                              | - Tivoli Access Manager Java runtime environment  
                              | - IBM Tivoli Directory Client  |
| Development system          | - IBM Global Security Kit (GSKit)  
                              | - Tivoli Access Manager runtime  
                              | - IBM Tivoli Directory Client  
                              | - Tivoli Access Manager application development kit  
                              | - Tivoli Access Manager Java runtime environment (required for developing Java applications)  
                              | - Tivoli Access Manager WebSEAL application development kit (required for developing Web Security applications) |
Table 34. Required components for Tivoli Access Manager system types (continued)

<table>
<thead>
<tr>
<th>Tivoli Access Manager System</th>
<th>Required Software Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization server</td>
<td>- IBM Global Security Kit (GSKit)</td>
</tr>
<tr>
<td></td>
<td>- Tivoli Access Manager runtime</td>
</tr>
<tr>
<td></td>
<td>- IBM Tivoli Directory Client</td>
</tr>
<tr>
<td></td>
<td>- Tivoli Access Manager authorization server</td>
</tr>
<tr>
<td>Policy proxy server</td>
<td>- IBM Global Security Kit (GSKit)</td>
</tr>
<tr>
<td></td>
<td>- Tivoli Access Manager runtime</td>
</tr>
<tr>
<td></td>
<td>- IBM Tivoli Directory Client</td>
</tr>
<tr>
<td></td>
<td>- Tivoli Access Manager policy proxy server</td>
</tr>
</tbody>
</table>

**Note:** If you use Microsoft Active Directory or Lotus Domino server as your user registry, the IBM Tivoli Directory Client is not required on Tivoli Access Manager systems in your secure domain.

**IBM Global Security Kit (GSKit)**

The two most common problems with the IBM Global Security Kit include:

- GSKit was not installed
- The wrong version of GSKit is installed (perhaps left over from a previous Tivoli Access Manager installation)

You can validate the correct installation of GSKit by checking its version as described in “IBM Global Security Kit (GSKit) version information” on page 6.

**LDAP server**

Communication between the IBM Tivoli Directory Client and the LDAP server can be tested using the `ldapsearch` command. This command also reveals the version of the LDAP server software. This command (entered as one line) can be executed from any machine with a copy of the IBM Tivoli Directory Client installed. The following command example, entered on one line, is appropriate when the LDAP server is configured for non-SSL communication:

```
ldapsearch -h ldapserver-hostname -p 389 -D "ldapadminDN" -w ldapadmin-password -b "" -s base objectclass="
```

**Note:** If this command fails on Windows system, check that the `ldapsearch` command being used is the one provided by the IBM Tivoli Directory Client.

The output of this command varies depending on which supported LDAP server you are using. The following output is generated by an IBM Tivoli Directory Server Version 5.2:

```
namingcontexts=CN=SCHEMA
namingcontexts=CN=LOCALHOST
namingcontexts=CN=PWDPOLICY
namingcontexts=CN=IBMPOLICIES
namingcontexts=O=IBM,C=US
namingcontexts=SECAUTHORITY=DEFAULT
namingcontexts=CN=CONFIGURATION
ibm-configurationnamingcontext=CN=CONFIGURATION
subschemasubentry=cn=schema
supportedextention=1.3.18.0.2.12.1
supportedextention=1.3.18.0.2.12.3
supportedextention=1.3.18.0.2.12.5
supportedextention=1.3.18.0.2.12.6
```

Chapter 9. Basic troubleshooting 87
Alternatively, if you have SSL configured for your LDAP server, you can test the SSL connection using the following command (entered as one line):

```
ldapsearch -h ldapserver-hostname -p 636 -D "*ldapadminDN"
-w ldapadmin-password -Z -K client-keyfile
-P key-password -b ** -s base objectclass**
```

The output should be similar to that listed above.

**IBM Tivoli Directory Client**

The verification procedure described above for the LDAP server used the IBM Tivoli Directory Client that had been installed on the LDAP server system itself. However, unless you are using Active Directory or Lotus Domino server for your Tivoli Access Manager user registry, each Tivoli Access Manager system requires the installation of the IBM Tivoli Directory Client, not just the machine with the LDAP server installed.
The verification procedure described above for the LDAP server, is also appropriate for verifying the functionality of the IBM Tivoli Directory Client on each Tivoli Access Manager system.

**Microsoft Active Directory**

**Client/server configuration**

**Server**

Start the MMC for Active Directory using the following menus:

Start > Program > Administrative Tools > Active Directory Users and Computers

If the Active Directory management console started successfully and you can browse all the objects in Active Directory, the Active Directory server has completed its configuration correctly. Otherwise, you can perform the procedures for unconfiguration and reconfiguration of the Active Directory described in either the *IBM Tivoli Access Manager for e-business Web Security Installation Guide* or the *IBM Tivoli Access Manager Base Installation Guide*.

**Client**

The client has to be configured into an existing Active Directory domain in order to perform Tivoli Access Manager configuration. To ensure that the client machine is part of the Active Directory domain, you can use System Properties to ensure the correct configuration of the client machine:

Start > Settings > Control Panel > System > System Properties

In the Control Panel, double click on System icon. The System Properties window appears. On the System Properties windows, click the Network Identification menu. If the Domain on the Network Identification contains the correct Active Directory domain, it indicates the client machine is properly configured into the Active Directory domain.

**Version number**

**Server**

Active Directory is included with Windows 2000 Advanced Server installation. Therefore, only one version of this software is possible. After successful Active Directory server configuration, the Active Directory server is started automatically during the reboot process.

**Client**

Active Directory is included with Windows 2000 Advanced Server installation. Therefore, only one version of this software is possible.

**Confirm client-server connection through ADSI**

Install “Windows 2000 Support Tools” in the SUPPORT\TOOLS directory on the Windows 2000 operating system CD. From that directory, start *setup.exe* and follow the installation guide to complete the “Windows 2000 Support Tools” installation.

Activate the ADSI Edit MMC window by starting:

Start > Program > Windows 2000 Support Tools > Tools > ADSI Edit

The user can set up the server connection by selecting:
Additionally, select the Advanced button on the Connection window to input the administrator ID and password to connect to the remote Active Directory server. If the connection is successful, the client machine will be able to communicate with the Active Directory server using ADSI.

**Lotus Domino Server**

**Client/server configuration**
For both the Notes client and the Domino server configuration, the user can refer to either *Domino 5 Administration Help* or *Notes 5 Help* documentation for instructions and information.

**Version number**

**Server**
To find out the Domino server version number, issue the following command on the Domino server console (or the command prompt on the server starting window)
```
show server
```

The list of Domino server information, including server version number, is displayed on the Domino server console.

**Client**
Refer to the *IBM Tivoli Access Manager for e-business Web Security Installation Guide* or *IBM Tivoli Access Manager Base Installation Guide* to install a Lotus Notes client on the client machine and configure it to the Domino server used for Tivoli Access Manager configuration. Upon completion of the Notes client configuration, the Notes GUI windows is displayed. Display the Notes client version number information from the Notes client information window by selecting the following GUI menus:
```
Help > About Notes
```

**Confirm client-server connection through the Notes APIs**
Once both the Notes client and Domino server have been configured successfully, the Notes client should be able to open any database in Domino server.

After starting the Notes client GUI, select the following menus to open any database in the Domino server:
```
File > Database > Open
```

From the pop-up *Open Database* window, you can select the correct Domino server and double-click on the database that you want to open. If the database in the selected Domino server opens, the Notes client is able to communicate with Domino server using Notes APIs.

**Tivoli Access Manager servers**
At a high level, you can determine which Tivoli Access Manager servers have been configured and which are running.

On UNIX platforms, use the following command:
```
# pd_start status
```
On Windows platforms, checked the Tivoli Access Manager entries that appear in the Services window. The Services icon can be found under Control Panel → Administrative Tools.

**Tivoli Access Manager policy server**

The `pdadmin` command can be used to verify the proper operation of the policy server. Enter the following command at the command line:

```
padmin -a sec_master -p sec_master-password
```

At the `pdadmin` prompt, enter the following commands (example output shown):

```
padmin> server list
webseald=cellname
(if WebSEAL is configured on the machine)

padmin> object list
/Management
/WebSEAL
(if WebSEAL is configured on the machine)

padmin> acl list
default-webseal
default-root
default-gso
default-management-proxy
default-policy
default-config
default-domain
default-management

default-replica

padmin> user list * 25
sec_master
ivmgrd/master
webseald/machinename
```

**Tivoli Access Manager authorization server**

The Tivoli Access Manager application development toolkit (ADK) includes a demonstration program named `authzn_demo(.exe)`. This program can be used, in remote mode, to validate the correct operation of the authorization server. Refer to the README file that accompanies this demonstration program for setup and execution instructions. The README file is located in the following directory:

`authzn-adk-install-dir/example/auth_demo/cpp`

**Tivoli Access Manager runtime**

The Tivoli Access Manager runtime may be installed on a runtime system with only GSKit and the IBM Tivoli Directory Client. In this case, the verification procedure for the Tivoli Access Manager runtime is the same as that described for the policy server in the section above.

**Tivoli Access Manager WebSEAL**

You can use a browser to verify that WebSEAL is operating properly. Enter the following URL into your browser:

```
https://webseal-machinename
```

Since a port number is not specified, it is assumed WebSEAL is listening on port 443 (HTTPS).
Your browser may give you the following warnings:

1. The certificate received from this Web server was issued by a company that you have not yet chosen to trust
2. The name within the certificate received from WebSEAL does not match the name of the system from which it was received

If these warnings occur, they simply indicate that you have not yet purchased your own server certificate for your WebSEAL server. Your browser is complaining that it has received a default server certificate from WebSEAL which contains default names for the issuing certificate authority and the name of the Web server.

Next, the browser prompts you to specify a Tivoli Access Manager user name and password. Enter sec_master for the user name and the password that you configured for sec_master during installation. If authentication is successful, an image labeled Tivoli Access Manager for WebSEAL appears.

**Junctioned third-party Web server**

Verifying the correct operation of a junctioned third-party Web application server is similar to that for WebSEAL itself (as described in the section above). Enter the following URL into your browser to verify that the third-party Web server is functioning properly:

http://junctioned-webserver-machinename

Since a port number is not specified, it is assumed the server is listening on port 80 (HTTP). If successful, the index.html page of the third-party Web server should appear.

The WebSEAL junction for the web server is created using a pdadmin command similar to the one in the following example (entered as one line). In this example, the junction point for the third-party Web server within the WebSEAL file space is named /myjunction:

```
pdadmin> server task webseald-webseal-machinename create -t tcp \ -p junctioned-server-port -h junctioned-webserver-machinename \ -c iv_user,iv_groups /myjunction
```

Try accessing the index.html page on the junctioned Web server, by accessing it through WebSEAL using the following URL:

https://webseal-machinename/myjunction

Your browser may again issue warnings about the WebSEAL server certificate that was received and prompts you again for a user name and password. Enter sec_master for the user name and the appropriate password. If successful, the index.html page of the third-party Web server appears.

**Tivoli Access Manager Plug-in for Web Servers**

You can use your browser to verify that Tivoli Access Manager Plug-in for Web Servers is operating properly.

Enter the following URL into your browser:

http://websorplugin-machinename
Since a port number is not specified, it is assumed that the Plug-in for Web Servers is listening on port 80 (HTTP).

Next your browser prompts you to specify a Tivoli Access Manager user name and password. Enter sec_master for the user name and the password that you configured for sec_master during installation. If successful, the default Web server page appears.
Chapter 10. Solutions to common problems

Before listing some of the common Tivoli Access Manager problems, it is worthwhile to mention that most common problems are the result of installation and configuration problems such as:

- Failure to install all of the software required by the Tivoli Access Manager product. This required software can include:
  - Operating system software
  - Operating system patches
  - Prerequisite software products
  - Prerequisite software product patches
- Failure to install the correct level of any of the software above
- Failure to install all of the required software components for any given type of Tivoli Access Manager system
- Failure to install or configure any of the above items properly
- Failure to adhere to all hardware prerequisites as well as disk and memory requirements

The information contained within the Tivoli Access Manager technical documentation is your best defense against the occurrence of any problems.

Section topics:
- “Tivoli Access Manager Base common problems” on page 95
- “LDAP common problems” on page 100
- “Tivoli Access Manager WebSEAL common problems” on page 101
- “Web Security common problems” on page 102
- “Upgrade common problems” on page 103

Tivoli Access Manager Base common problems

- “Unable to configure the policy server” on page 95
- “Unable to create new user” on page 97
- “Invalid account” on page 97
- “Insufficient disk space causes problems” on page 98
- “Unexpected permit/deny access to resources” on page 98

Unable to configure the policy server

Problem:
I am unable to configure the policy server.

Solution:
There are many possible causes for this problem. A few are listed below:

Suggestion 1:

The policy server may be unable to communicate with the configured user registry. Examine the following possible causes:
Verify that the user registry has not been stopped.

If using LDAP for your user registry, verify that the LDAP client on your machine can still communicate with the LDAP server. Issue an LDAP command such as the following (entered as one line) to learn whether the LDAP server is responsive:

```
ldapsearch -h ldapserver-hostname -p 389 -D "ldapadmin-DN" \
-w ldapadmin-password -b "" -s base objectclass**
```

**Note:** On Microsoft Windows systems: If this command fails, ensure that the `ldapsearch` command comes from the IBM Tivoli Directory Client.

Keep in mind that the output of this command can vary depending upon which supported LDAP server you are using. The command above assumes that the LDAP server has been configured to listen on port 389. Also verify that the user registry is still configured to communicate over the same port that was specified to Tivoli Access Manager during the configuration of the Tivoli Access Manager runtime.

**Suggestion 2:**

The configuration of the policy server can fail if Tivoli Access Manager is unable to create the `secAuthority=default` suffix on the IBM Directory server.

There are 3 steps required to prepare an LDAP server for use with Tivoli Access Manager. These steps must be completed prior to configuring Tivoli Access Manager:

1. **Apply the LDAP schema modifications required by Tivoli Access Manager to the IBM Directory Server.** This step is required for IBM Directory Server versions 4.1 and 5.1 only. IBM Tivoli Directory Server version 5.2, which is included with IBM Tivoli Access Manager, already includes these schema modifications.

   The schema modifications are contained in the file `secschema.def`, located in the common directory on the IBM Tivoli Access Manager Base CD. The schema modifications can be applied to the IBM Directory Server using either the `ivrgy_tool` command:

   `ivrgy_tool -d -h ldap_host -p port -D ldap_admin -w pwd schema`

   (see the [IBM Tivoli Access Manager for e-business Command Reference](#) for more information)

   or the `ldapmodify` command:

   `ldapmodify -v -h ldap_host -p port -D ldap_admin -w pwd -P /tmp/secschema.def`

   **Note:** The `apply_ldap41_patch` file, which was provided in the previous version of Tivoli Access Manager, is no longer shipped with the product.

2. **Create the `secAuthority=default` suffix.**

3. **Stop and restart the IBM Directory Server to enable the IBM Directory server to recognize the newly created suffix.**

When the easy installation wizard is used, these 3 steps are performed automatically and are transparent to the user. When native install is used, these steps must be performed manually.

Failure to perform these steps prior to configuring the policy server results in a configuration failure.
Suggestion 3:

If you are using a Lotus Domino server for the Tivoli Access Manager user registry and the configuration of your policy server fails, ensure that you have installed and configured the Lotus Notes client on your system and that the file NNOTES.DLL can be found in your system path. If NNOTES.DLL cannot be found in the system path, explicitly add it to the system path.

Unable to create new user

Problem:
I am having difficulty creating a new Tivoli Access Manager user.

Solution:
One of the most common error messages seen when creating a new user is:
Could not perform the administration request.
Error: Password rejected due to the Minimum Non-Alphabetic Characters policy (status 0x13212131)

This error indicates, for example, that the password "abc" specified when attempting to create the new user does not comply with one of the "user" password policies currently defined to Tivoli Access Manager. To view the help text associated with the Tivoli Access Manager policy commands, enter the following command:
pdadmin> help policy

The password policy error above can be solved using one of the solutions below:

- Determine the minimum non-alphabetic character policy currently defined for Tivoli Access Manager:
pdadmin> get min-password-non-alphas

Then create the user using a password that contains the required minimum of non-alpha characters:

- Modify the Tivoli Access Manager non-alphabetic character policy prior to creating the user:
pdadmin> set min-password-non-alphas number

Invalid account

Problem:
I am unable to authenticate with a Tivoli Access Manager user identity I have recently created.

Solution:
Tivoli Access Manager user definitions are initially created with the attribute:
Account valid = no

This condition is frequently the cause of authentication failures. Use the following command to change the user’s Account valid attribute to "yes":
pdadmin> user modify user-name account-valid yes

Use the following command to verify that the change was made successfully:
pdadmin> user show user-name
**Insufficient disk space causes problems**

**Problem:**
Insufficient disk space can cause problems with the proper functioning of Tivoli Access Manager.

**Solution:**
Ensure that adequate disk space exists on whatever drives (Windows) or file systems (UNIX) you used to install Tivoli Access Manager.

On the Windows platforms, concerns for adequate disk space includes the drives which contain the following directories:

- The Tivoli Access Manager installation directory
- The WebSEAL installation directory
- The Tivoli Access Manager Plug-in for Web Servers installation directory
- The Tivoli Access Manager Plug-in for Edge Server installation directory

On UNIX platforms, concerns for adequate disk space includes the file systems that contain the `/opt` and the `/var` directories. Use the `df -k` command to display the free disk space for each file system. The `-k` option causes the disk space to be displayed in kilobytes.

**Unexpected permit/deny access to resources**

**Problem:**
Accesses to a protected system resource are either being unexpectedly granted or denied.

**Solution:**
It is always wise to first validate that the Tivoli Access Manager processes are started and running normally. Also check to ensure that the Tivoli Access Manager message log files do not flag any operational problems. If Tivoli Access Manager seems operationally sound, the problem is likely due to the policies that have been defined and applied to that system resource.

There are three Tivoli Access Manager policy mechanisms that can be used to control access to your protected system resources: ACLs, POPs, and authorization rules. Use the `pdadmin` command to learn which ACL within your protected object space hierarchy controls access to the protected resource in question.

If an ACL is directly attached to the protected object in question, then it defines the ACL policy for that object. If an ACL is not directly attached to the protected object in question, then the controlling ACL is the nearest one above in the protected object hierarchy.

The following command lists all ACLs which are defined in Tivoli Access Manager:

```
padmin> acl list
```

The following command enables you to learn where each of those ACLs is attached within the protected object space hierarchy.

```
padmin> acl find acl-name
```

Examine the controlling ACL using the following command to ensure it is correct for the type of enforcement desired. Correct the ACL definition if needed:
pdadmin> acl show acl-name

Tivoli Access Manager access control depends on two conditions:

- The ACL that controls the requested object must contain appropriate access permissions for the requesting user.
- The requested object must be accessible to the requesting user.

Accessibility to protected objects is controlled by the traverse (T) permission. The traverse permission is only applied to container objects in the protected object space. The traverse permission specifies that a user, group, any-other, or unauthenticated user, identified in the ACL entry, has permission to pass through this container object in order to gain access to a protected resource object below in the hierarchy.

A protected object is accessible to a requester if the requester possesses the traverse permission on each ACL attached to container objects above the requested resource on the path towards root and including root.

Additionally, use the `pdadmin` command to learn which POP (if any) within your protected object space hierarchy controls access to the protected resource in question.

If a POP is directly attached to the protected object in question, then it defines the POP policy for that object. If a POP is not directly attached to the protected object in question, then the controlling POP is the nearest one above in the protected object hierarchy.

The following command lists all of the POPs which are defined for Tivoli Access Manager:

```
padmin> pop list
```

The following command enables you to learn where a particular POP is attached within the protected object space hierarchy:

```
padmin> pop find pop-name
```

Examine the controlling POP using the following command to ensure it is correct for the type of enforcement desired. Correct the POP definition if needed.

```
padmin> pop show pop-name
```

If an authorization rule is directly attached to the protected object in question, then it defines the rule policy for that object. If an authorization rule is not directly attached to the protected object in question, then the controlling rule is the nearest one above in the protected object hierarchy.

The following command lists all of the authorization rules defined for Tivoli Access Manager:

```
padmin> authzrule list
```

The following command enables you to learn where a particular authorization rule is attached within the protected object space hierarchy:

```
padmin> authzrule find authzrule-name
```

Use the following command to examine the controlling authorization rule and ensure it is correct for the type of enforcement required. Correct the rule definition if needed.
LDAP common problems

This section details common problems encountered when using an LDAP-based user registry, such as IBM Tivoli Directory Server.

- “LDAP does not start after secAuthority=Default suffix is created” on page 100
- “Insufficient LDAP access privileges to perform operation” on page 101
- “IBM Tivoli Directory Server error log warnings” on page 101

LDAP does not start after secAuthority=Default suffix is created

Problem:
IBM Tivoli Directory Server does not start after the secAuthority=Default suffix is created.

Solution:
When using IBM Directory Server versions 4.1 or 5.1, the user might experience a failure where the IBM Directory Server cannot be restarted after the secAuthority=Default suffix has been created.

There are 3 steps required to prepare an LDAP server for use with Tivoli Access Manager. These steps must be completed prior to configuring Tivoli Access Manager:

1. Apply the LDAP schema modifications required by Tivoli Access Manager to the IBM Directory Server. This step is required for IBM Directory Server versions 4.1 and 5.1 only. IBM Tivoli Directory Server version 5.2, which is included with IBM Tivoli Access Manager, already includes these schema modifications.

   The schema modifications are contained in the file secschema.def, located in the common directory on the IBM Tivoli Access Manager Base CD. The schema modifications can be applied to the IBM Directory Server using either the ivrgy_tool command:
   
   ```
   ivrgy_tool -d -h ldap_host -p port -D ldap_admin -w pwd schema
   ```
   
   (see the IBM Tivoli Access Manager for e-business Command Reference for more information)

   or the ldapmodify command:
   
   ```
   ldapmodify -v -h ldap_host -p port -D ldap_admin -w pwd -P /tmp/sec_schema.def
   ```

   **Note:** The apply_ldap41_patch file, which was provided in the previous version of Tivoli Access Manager, is no longer shipped with the product.

2. Create the secAuthority=default suffix.

3. Stop and restart the IBM Directory Server to enable the IBM Directory server to recognize the newly created suffix.

When the easy installation wizard is used, these 3 steps are performed automatically and are transparent to the user. When native install is used, these steps must be performed manually.
If the user attempts to create the `secAuthority=default` suffix and restart the IBM Directory Server prior to applying the schema modifications required by Tivoli Access Manager, the IBM Directory Server will fail to restart.

When the server fails to restart it will log an error message in the `slapd.errors` file indicating that the `secAuthority` attribute is not defined. The `slapd.errors` file is located in the `/tmp` directory on UNIX systems and in the `ldap-install-dir\tmp` directory on Windows systems, where `ldap-install-dir` is the directory where IBM Tivoli Directory Server was installed.

**Insufficient LDAP access privileges to perform operation**

**Problem:**
I am experiencing an error which reads:

```
Insufficient LDAP access privileges to perform operation.
```

**Solution:**
This message indicates that a supplied LDAP suffix does not have the correct ACLs attached. Possible reasons for this are:

- During configuration, Tivoli Access Manager was unable to attach ACLs to the existing suffix because the directory entries necessary to instantiate the suffix had not been created
- The suffix was added after the initial configuration of the Tivoli Access Manager management server and the appropriate ACLs were not added manually as is required

The Web Administration Tool provided with the IBM Tivoli Directory client can be used to check the suffix and add the appropriate ACLs manually. For information on how to accomplish this, see the *IBM Tivoli Access Manager Base Administration Guide*.

Fix the ACLs on the suffix and then retry the failing command.

**IBM Tivoli Directory Server error log warnings**

**Problem:**
IBM Tivoli Directory Server error log indicates several "does not exist" warnings.

**Solution:**
When the Tivoli Access Manager policy server is configured, the policy server is first unconfigured as part of this configuration process to ensure that it has been cleaned up completely.

The unconfiguration step attempts to remove entries in the LDAP registry.

When the policy server has not yet completed configuration, these entries have not yet been created and may not exist in the LDAP registry. The IBM Tivoli Directory Server logs these entry removal attempts as warnings in its error log. These warnings are therefore normal and can be safely ignored.

**Tivoli Access Manager WebSEAL common problems**

- "WebSEAL does not respond on ports 80 or 443" on page 102
- "Multiple logins required during e-community" on page 102
WebSEAL does not respond on ports 80 or 443

Problem:
WebSEAL does not respond on either port 80 or port 443.

Solution:
Investigate whether you have another Web server installed on the WebSEAL system. For example, the IBM HTTP Server is commonly installed as part of the installation of an IBM Tivoli Directory Server. If another Web server is installed on the same system as WebSEAL, reconfigure it to listen upon ports other than those configured for use by WebSEAL.

Multiple logins required during e-community

Problem:
WebSEAL e-community users are prompted to login more than once.

Solution:
This problem can occur if two WebSEAL servers are configured in the same domain—one configured as the Master Authentication Server (MAS) and the other configured to use the MAS server for authentication. Attempts to access the latter may require the user to login more than once if the difference in system times between the two WebSEAL servers is too great. Ensure that the system time on each WebSEAL server participating in an e-community is synchronized.

Active Directory password complexity rules enforced even when disabled

Problem:
When Microsoft Windows 2003 Active Directory is used as the user registry for Tivoli Access Manager, changing a password using WebSEAL might return an error if the password does not follow the Microsoft account password complexity requirements policy. This error can occur even if the Active Directory Domain account password policy "Password must meet complexity requirements" is set the "Disabled".

Solution:
Ensure that your Tivoli Access Manager password meets the Microsoft Active Directory password complexity requirements when Microsoft Active Directory is used as the user registry. These requirements can be found at the following Web location: [http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/windowsserver2003/proddocs/entserver/504.asp](http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/windowsserver2003/proddocs/entserver/504.asp)

Web Security common problems

Troubleshooting information for the Web Security components can be found in the corresponding Integration Guide:

- IBM Tivoli Access Manager for e-business IBM WebSphere Application Server Integration Guide
- IBM Tivoli Access Manager for e-business BEA WebLogic Server Integration Guide
- IBM Tivoli Access Manager for e-business IBM WebSphere Edge Server Integration Guide
- IBM Tivoli Access Manager for e-business Plug-in for Web Servers Integration Guide
Upgrade common problems

- "Cannot create users and groups after upgrade" on page 103

Cannot create users and groups after upgrade

**Problem:**
Tivoli Access Manager does not have authority to create users and groups after upgrading the IBM Tivoli Directory server.

**Solution:**
When the IBM Tivoli Directory Server component is chosen as the user registry and you are upgrading from a previous version of Tivoli Access Manager, the IBM Tivoli Directory Server component must be migrated first, if all components are located on the same machine.

The migration of the IBM Tivoli Directory server is performed by following the instructions in the *IBM Tivoli Access Manager Upgrade Guide*.

These instructions guide you through the process of backing up the current data with the `db2ldif` utility, upgrading the IBM Tivoli Directory Server, and restoring the data with the `bulkload` utility.

When you use the `bulkload` utility, you should specify the `–A yes` option to have it properly process Access Control List (ACL) updates. If the ACLs are not loaded properly, Tivoli Access Manager will not have the authority to perform the needed tasks to create and maintain user and group information.

If bulkload fails to update the ACLs properly and these symptoms occur, you can create the ACLs manually by following the “Applying Access Manager ACLs to new LDAP suffixes” procedure in the “Using the LDAP registry” chapter of the *IBM Tivoli Access Manager Base Administration Guide*. Apply the ACLs to all existing LDAP suffixes and `secAuthority=Default` entries below all defined users in the LDAP server using the Web Administration Tool. This will restore the proper authority to allow Tivoli Access Manager to continue.
Chapter 11. Contacting customer support

If you encounter a problem with the Tivoli Access Manager product that cannot be resolved using the problem determination information provided in this guide, contact IBM Tivoli Software Support. The proper method for contacting support is described in the Customer Support Handbook, which is located at the following Web site: [http://techsupport.services.ibm.com(guides/handbook.html](http://techsupport.services.ibm.com/guides/handbook.html)

Throughout this guide, you are directed to collect certain kinds of information to assist Customer Support in finding a solution to your problem. Collect this information, but do not send it to Customer Support until you are directed to do so. The procedures and methods for transmitting such data undergo periodic change; it is best to get the latest instructions from Customer Support.

General data to collect for customer support

The following is a list of the types of data that should be collected and made available to Customer Support should they request it:

- Brief description of the class of problem, such as install, configuration, audit, system failure, or performance
- Hardware configuration (machine make and model number), operating system type, version number, patch levels
- Language/locale information
- Tivoli Access Manager level, including any fix packs and patches.
- Tivoli Access Manager and LDAP server—the systems on which they are located, network connectivity to these systems, version number (including fixpacks) of the servers
- Are any of the systems (Tivoli Access Manager server, LDAP server) configured with multiple IP addresses?
- Detailed description of the problem, whether the problem can be recreated, and if so, the steps required to recreate the problem
- Output from the pdversion utility
- Output from the pbackup utility
- Collect any trace logs which have been generated
  Typically trace logs will be generated only at the request of IBM customer support personnel.
- Timeframe in which the problem occurred (to relate it back to the log entries)

Collecting defect information for pdadmin and C APIs

This section describes how to collect problem determination information when reporting a defect on the Tivoli Access Manager administration C and authorization C APIs and the pdadmin command line interface.

Basic procedure

- Turn on tracing on the machine that is running the affected Tivoli Access Manager server (could be pdmgrd and/or pdacld depending on the API being used).
- Restart pdmgrd and/or pdacld.
- Turn on tracing on the machine where you are executing `pdadmin` or the application/test case that invokes the C APIs (administration C API or authorization C API).
- Restart `pdadmin` or the application/test case that invokes the C APIs.
- Execute the operations that failed.
- Use the `pdbackup` utility, with the appropriate information list file, to collect the trace log files (in addition to other system data). You can also follow steps listed below to collect the trace files individually.

**Note:** One way to turn tracing on is to edit the `base-install-dir\etc\routing` file and uncomment the last entry in this file. You can also turn tracing on dynamically for authorization API applications, but this approach will not work for `pdmgrd` or for applications that use only the administration C API and not the authorization C API.

### Getting the pdmgrd trace file

```bash
# cd base-install-dir\log
# cat ivmgrd.pid
```

(Make note of the `pdmgrd` process ID from the output of this command)

Collect the `pdmgrd-process-id.trace.log.*` files

### Getting the pdacld trace file

```bash
# cd base-install-dir\log
# cat ivacld.pid
```

(Make note of the `pdacld` process ID from the output of this command)

Collect the `pdacld-process-id.trace.log.*` files

### Getting the pdmgrproxyd trace file

```bash
# cd base-install-dir\log
# cat pdmgrproxyd.pid
```

(Make note of the `pdmgrproxyd` process ID from the output of this command)

Collect the `pdmgrproxyd-process-id.trace.log.*` files

### Getting the pdadmin and C API trace file

Note down the process ID of the `pdadmin` process or the process ID of the application/test case that invokes the C API.

```bash
# cd base-install-dir\log
```

Collect the `process-id.trace.log.*` files

### Getting the message files

Collect the following files from the `log` directory on all the machines that run Tivoli Access Manager servers and applications:

```bash
msg_notice*log
msg_warning*log
msg_error*log
msg_fatal*log
```

106 IBM Tivoli Access Manager for e-business: Problem Determination Guide
Collecting defect information for Java APIs

This section describes how to collect problem determination information when reporting a defect on the Tivoli Access Manager administration Java and authorization Java APIs.

Basic procedure

- Turn on tracing on the machine that is running the affected Tivoli Access Manager server (could be pdmgrd and/or pdacld depending on the API being used).
- Restart pdmgrd and/or pdacld.
- Turn on tracing on the machine where you are executing the application/test case that invokes the Java APIs (administration Java API or authorization Java API).
- Restart the application/test case that invokes the Java APIs.
- Execute the operations that failed.
- Use the pdbackup utility, with the appropriate information list file, to collect the trace log files (in addition to other system data). You can also follow steps listed below to collect the trace files individually.

Note: Edit the pdmgrd_routing, pdacld_routing, and pdmgrproxyd_routing files to turn tracing on for the policy server, authorization server, and policy proxy server, respectively. To turn on tracing in applications that use the authorization Java API or the administration Java API, refer to “Enabling tracing and messaging for Java runtime environment components,” or to “Enabling trace logging for the Java runtime environment” on page 54.

Enabling tracing and messaging for Java runtime environment components

Edit the properties file associated with the Java application and enable tracing:

```
baseGroup.PDJapp_nameTraceLogger.isLogging = true
```

Getting the pdmgrd trace file

```
# cd base-install-dir/log
# cat ivmgrd.pid
```

(Make note of the pdmgrd process ID from the output of this command)

Collect the pdmgrd-process-id.trace.log.* files.

Getting the pdacld trace file

```
# cd base-install-dir/log
# cat ivacld.pid
```

(Make note of the pdacld process ID from the output of this command)

Collect the pdacld-process-id.trace.log.* files.

Getting the pdmgrproxyd trace file

```
# cd base-install-dir/log
# cat pdmgrproxyd.pid
```
(Make note of the `pdmgrproxyd` process ID from the output of this command)

Collect the `pdmgrproxyd-process-id.trace.log.*` files

**Getting the Java API trace files**

```
# cd base-install-dir/log
```

Collect the `trace_app_name.log` files.
Appendix A. Example routing file

Here is a sample of the routing file for Tivoli Access Manager Base from a Microsoft Windows system.

```
# #
# The source code for this program is not published or otherwise divested
# of its trade secrets, irrespective of what has been deposited with the
# U.S. Copyright Office.
#
# COPYRIGHT NOTICE
# ALL RIGHTS RESERVED (OCE).
#
# (c) Copyright IBM Corp. 1994-2003. All Rights Reserved.
# # # #

## This is a prototype Access Manager serviceability routing file.
## Leading whitespace is ignored, as are lines whose first non-whitespace
## character is a #. As installed, all lines in this file are commented
## out. The PD_SVC_ROUTING_FILE environment variable may be used to specify
## this file or an alternate routing file for the application. Otherwise
## it must be located in the etc subdirectory of the Access Manager installation.
## (It is not an error if PD_SVC_ROUTING_FILE points to a non-existent file.)

## This file is consulted if no switch were given on the command-line, or
## if no environment variable (SVC_level or SVC_comp_DBG) could be found.

## Production messages are parsed as:
## level:where:parameter
## level is FATAL ERROR WARNING NOTICE NOTICE_VERBOSE or * meaning all.
## where is STDERR STDOUT FILE (or TEXTFILE) UTF8FILE
## XMLSTDERR XMLSTDOUT XMLFILE DISCARD
## parameter is the filename where "%ld" becomes the process-id.

## UTF8FILE are equivalent to TEXTFILE with the exception that the
## data will be encoded in UTF-8.
## For example:
#ERROR:UTF8FILE.C:/PROGRA~1/Tivoli/POLICY~1/log/msg_error.log
#*:*.9:UTF8FILE.10.1000:C:/PROGRA~1/Tivoli/POLICY~1/log/trace_%ld.log;STDOUT:-
## XMLSTDERR, XMLSTDOUT, and XMLFILE are equivalent to
## STDOUT, STDOUT, and TEXTFILE with the exception that the
## data will be written in "Log XML" format and encoded in UTF-8.
## For example:
#ERROR:XMLSTDERR::XMLFILE.C:/PROGRA~1/Tivoli/POLICY~1/log/msg_error.log
#*:*.9:XMLFILE.10.1000:C:/PROGRA~1/Tivoli/POLICY~1/log/trace_%ld.log;XMLSTDOUT:-

## If FILE ends with ".n.m" then at most "n" files of "m" entries
## each will be written where ".n" will be appended to each generation of
## file. So to keep the last 1000 warnings around in 10 files for all
## programs:
#NOTICE:FILE.10.1000:C:/PROGRA~1/Tivoli/POLICY~1/log/msg__syslog.log

## The routing of debug messages is specified like this:
## comp:level,level,level:where:parameter
## "comp" is the component (acl, myr, las, etc.)
## "level" is a list of sub-component levels for each component
## "where" and "parameter" are as above.
## Sub-components are listed in the same file associated with the component.
```
Levels are subcomp.n, where n is 1 to 9. They are parsed in order, so put sub-component wildcard entries first.

A component cannot be named twice -- component debug messages all go into one place.

So ACL tracing might be:

```
#acl:*.1,mgmt.9:STDOUT:-
```

Which sets low-level debugging of everything, and full debugging for the management part.

Each component can have its own entry. To turn on parts of MGR use something like the following:

```
#mgr:objmgmt.9,srmgmt.4:STDOUT:
```

Other examples:

```
#acl:*.2:FILE:C:/PROGRA~1/Tivoli/POLICY~1/log.trace__acl.log
#mgr:*.9:FILE:C:/PROGRA~1/Tivoli/POLICY~1/log.trace__mgr.log
```

A component cannot be named twice -- component debug messages all go into one place.

Environment variables can be present in the filenames and will be expanded at runtime.

Special environment variable PDDIR will be set at runtime to the Policy Director installation directory.

```
FATAL:STDERR:-;FILE:C:/PROGRA~1/Tivoli/POLICY~1/log/msg__fatal.log
ERROR:STDERR:-;FILE:C:/PROGRA~1/Tivoli/POLICY~1/log/msg__error.log
WARNING:STDERR:-;FILE:C:/PROGRA~1/Tivoli/POLICY~1/log/msg__warning.log
NOTICE:FILE:C:/PROGRA~1/Tivoli/POLICY~1/log/msg__notice.log
#NOTICE_VERBOSE:STDERR:-;FILE:C:/PROGRA~1/Tivoli/POLICY~1/log/msg__verbose.log
```

Routing Entries for the Trace Output

```
# Route to STDOUT of each process
#*:*.9:STDOUT:-
```

```
# Route to a per-process text file
#*:*.9:TEXTFILE.10.10000:C:/PROGRA~1/Tivoli/POLICY~1/log/trace__%ld.log
```
Appendix B. Sample Java application properties file

Here is an example of the contents of a properties file from a Java application called PDG1-svr. This file is used to set the properties associated within that particular Java application. If a property is not defined in this file, the $JAVA_HOME/PolicyDirector/PDJLog.properties file associated with the JRE is used. (See Appendix C, “Sample PDJLog.properties file,” on page 115 for a sample.)

Note: The lines in the properties file shown here has been re-ordered and comments have been added to annotate major sections. The actual properties file produced by the com.tivoli.pd.jcfg.SvrSslCfg command is unordered and without comments as it is produced programmatically.

```
#Config last generated on
#Thu Oct 09 14:08:25 CDT 2003
#
# Message handling
#
baseGroup.PDJPDG1-svrMessageLogger.parent=PDJMessageLogger
baseGroup.PDJPDG1-svrMessageLogger.component=PDG1-svr
baseGroup.PDJPDG1-svrMessageLogger.name=PDJPDG1-svrMessageLogger
baseGroup.PDJPDG1-svrMessageLogger.description=Access Manager PDG1-svr Message Logger
baseGroup.PDJPDG1-svrMessageLogger.handlerNames=PDJPDG1-svrMessageFileHandler
baseGroup.PDJPDG1-svrMessageLogger.filterNames=PDJPDG1-svrMessageAllMaskFilter PDJPDG1-svrMessageClassFilter
baseGroup.PDJPDG1-svrMessageLogger.isLogging=true
#
baseGroup.PDJPDG1-svrMessageFileHandler.description=Access Manager PDG1-svr Message File Handler
baseGroup.PDJPDG1-svrMessageFileHandler.fileName=C:\Program Files\ibm\tivoli\common//logs/msg__PDG1-svr.log
baseGroup.PDJPDG1-svrMessageFileHandler.parent=PDJMessageFileHandler
#baseGroup.PDJPDG1-svrMessageFileHandler.maxFileSize=
#baseGroup.PDJPDG1-svrMessageFileHandler.maxFiles=
#baseGroup.PDJPDG1-svrMessageAllMaskFilter.mask=FATAL | ERROR | WARNING | NOTICE | NOTICE_VERBOSE
baseGroup.PDJPDG1-svrMessageAllMaskFilter.parent=PDJMessageAllMaskFilter
#
baseGroup.PDJPDG1-svrMessageClassFilter.className=com.tivoli.pd.jras.pdjlog.PDJClassFilter
baseGroup.PDJPDG1-svrMessageClassFilter.className=com.tivoli.pd.jras.pdjlog.PDJClassFilter
#
```

Figure 17. Sample application-specific properties file (1 of 3)
# Trace handling

`baseGroup.PDJPDG1-svrTraceLogger.parent=PDJTraceLogger`
`baseGroup.PDJPDG1-svrTraceLogger.component=PDG1-svr`
`baseGroup.PDJPDG1-svrTraceLogger.name=PDJPDG1-svrTraceLogger`
`baseGroup.PDJPDG1-svrTraceLogger.description=Access Manager PDG1-svr Trace Logger`
`baseGroup.PDJPDG1-svrTraceLogger.handlerNames=PDJPDG1-svrTraceFileHandler`
`baseGroup.PDJPDG1-svrTraceLogger.filterNames=PDJPDG1-svrTraceAllMaskFilter PDJPDG1-svrTraceClassFilter`
`baseGroup.PDJPDG1-svrTraceLogger.isLogging=false`

`# baseGroup.PDJPDG1-svrTraceAllMaskFilter.mask=9`
`baseGroup.PDJPDG1-svrTraceAllMaskFilter.parent=PDJTraceAllMaskFilter`

`# baseGroup.PDJPDG1-svrTraceClassFilter.className=com.tivoli.pd.jras.pdjlog.PDJClassFilter`
`baseGroup.PDJPDG1-svrTraceClassFilter.description=Access Manager PDG1-svr Trace Class Filter`

`# baseGroup.PDJPDG1-svrTraceFileHandler.description=Access Manager PDG1-svr Trace File Handler`
`baseGroup.PDJPDG1-svrTraceFileHandler.parent=PDJTraceFileHandler`
`baseGroup.PDJPDG1-svrTraceFileHandler.fileName=C:\Program Files\ibm\tivoli\common//logs/trace__PDG1-svr.log`
`#baseGroup.PDJPDG1-svrTraceFileHandler.maxFileSize=
#baseGroup.PDJPDG1-svrTraceFileHandler.maxFiles=

# Tivoli Common Directory
#
tcd_enabled=true
tcd_home=C:\\PROGRA~1\tivoli_common_dir=C:\\Program Files\ibm\tivoli\common

Figure 18. Sample application-specific properties file (2 of 3)
appsvr-authzsvrs=PDG1:7136:1;
appsvr-configname=PDG1
appsvr-credcache-life=7200
appsvr-dbdir=
appsvr-domain=Default
appsvr-host=pdg1
appsvr-listen=false
appsvr-mode=remote
appsvr-plcysvrs=PDG1:7135:1;
appsvr-port=7135
appsvr-servername=PDG1-svr
appsvr-username=PDG1/svr
client.deliveryPolicy=com.tivoli.pd.jaudit.client.AMAuditClientDeliveryPolicy
client.doAudit=false
client.serverHost=localhost
client.serverPort=7131
client.source=localSource
config_type=full
jar-files=c:\\program files\\ibm\\java131\\jre\\lib\\ext\\PD.jar
java-home=c:\\program files\\ibm\\java131\\jre
local_domain=Default
mgmt_domain=Default
pd-home=C:\\PROGRA~1\\Tivoli\\POLICY\\1
pdcert-dn=cn=PDG1/pdg1, cn=SecurityDaemons, secAuthority=Default
pdcert-pw=U5+jTtUJP3qVhIcbajNC11GChbhzS1j2DNq6xuyldY6q078QyvP3cKZg8buf1nDK+XZQaA/Zcj9f4h/\nngdJ
pdcert-url=file://c:/tmp/pd/pdadmin.ks
pdjlog_cfg_file=
PDJPDG1-svrMessageClassFilter=
pdv==home=C:\\PROGRA~1\\Tivoli\\POLICY\\1
product-code=
tcd_enabled=true
tcd_home=C:\\PROGRA~1
vivo1_common_dir=C:\\Program Files\\ibm\\tivoli\\common
version=510

Figure 19. Sample application-specific properties file (3 of 3)
Appendix C. Sample PDJLog.properties file

Here is an example of the contents of the default $JAVA_HOME/PolicyDirector/PDJLog.properties file. This file is used to set properties for a Java application if that property is not defined in the application-specific properties file, or if the application-specific properties file is inaccessible or does not exist.

```ini
# This section shows the key/value pairs that may be specified to configure a PDJTraceAllMaskFilter. The mask key determines the level at which trace is captured. The valid trace levels are one of the numerals 1-9. The trace levels are nested. Specifying a value of 9 includes levels 1-8, specifying a value of 7 includes levels 1-7, and so on.
# To set the same mask for all the PDJRTE components:
# (1) Set the mask for the PDJTraceAllMaskFilter to the desired mask, and (2) comment out below the AllMaskFilter entries for the individual components such as the PDJadminAllMaskFilter.
#-------------------------------------------------------------
baseGroup.PDJTraceAllMaskFilter.mask=9
```

```ini
# This section shows the key/value pairs that may be specified to configure a PDJMessageLogger. The only value that should potentially be modified for this section is the isLogging value.
# To turn trace on for all the PDJRTE components: (1) Set the isLogging value for the PDJTraceLogger to true and (2) comment out below the isLogging value entry for the individual PDJRTE component trace loggers such as the PDJadminTraceLogger.
#-------------------------------------------------------------
baseGroup.PDJTraceLogger.isLogging=false
```

```ini
# This section shows the key/value pairs that may be specified to configure a PDJMessageLogger. The only value that should potentially be modified for this section is the isLogging value, which is set to true by default. To turn on messaging for individual handlers attached to this logger, set the isLogging value for each desired handler, such as PDJNoticeFileHandler.
#-------------------------------------------------------------
baseGroup.PDJMessageLogger.isLogging=true
```

```ini
# This section shows the key/value pairs that may be specified to configure an PDJTraceFileHandler.
# The fileName key specifies the fixed part of the name of the file to which the trace output for all the PDJRTE components is written out.
#-------------------------------------------------------------
```

© Copyright IBM Corp. 2002, 2003
The first file will have the base part of this name with the numeral 1 appended. Each subsequent file (controlled by the maxFiles and maxFileSize keys below) will have their respective numbers appended to the base part of the name. For instance, if maxFiles=3 and maxFileSize=10, there will be 3 files of 10 blocks (512 bytes each) written before rollover occurs.

The default fileName is trace__amj.log. The default maxFiles is 3. The default maxFileSize is 512 blocks.

The fully-qualified file names for the default keys are:

For Windows
- <pd-home>/log/trace__amj1.log
- <pd-home>/log/trace__amj2.log
- <pd-home>/log/trace__amj3.log

For unix
- <var-dir>/PolicyDirector/log/trace__amj1.log
- <var-dir>/PolicyDirector/log/trace__amj2.log
- <var-dir>/PolicyDirector/log/trace__amj3.log

If either PDJLog.properties or PD.properties is not found, no logging will take place.

This section shows the key/value pairs that may be specified to configure a PDJFatalFileHandler.

The fileName key specifies the fixed part of the name of the file to which the fatal error output for all the PDJRTE components is written out. The first file will have the base part of this name with the numeral 1 appended. Each subsequent file (controlled by the maxFiles and maxFileSize keys below) will have their respective numbers appended to the base part of the name. For instance, if maxFiles=3 and maxFileSize=10, there will be 3 files of 10 blocks (512 bytes each) written before rollover occurs.

The default fileName is msg__amj_fatal.log. The default maxFiles is 3. The default maxFileSize is 512 blocks.

The fully-qualified file names for the default keys are:

For Windows
- <pd-home>/log/msg__amj_fatal1.log
- <pd-home>/log/msg__amj_fatal2.log
- <pd-home>/log/msg__amj_fatal3.log

For unix
- <var-dir>/PolicyDirector/log/msg__amjFatal1.log
- <var-dir>/PolicyDirector/log/msg__amjFatal2.log
- <var-dir>/PolicyDirector/log/msg__amjFatal3.log

If either PDJLog.properties or PD.properties is not found, no logging will take place.
This section shows the key/value pairs that may be specified to configure a PDJErrorFileHandler.

The fileName key specifies the fixed part of the name of the file to which the error output for all the PDJRTE components is written out. The first file will have the base part of this name with the numeral 1 appended. Each subsequent file (controlled by the maxFiles and maxFileSize keys below) will have their respective numbers appended to the base part of the name. For instance, if maxFiles=3 and maxFileSize=10, there will be 3 files of 10 blocks (512 bytes each) written before rollover occurs.

The default fileName is msg__amj_error.log.
The default maxFiles is 3
The default maxFileSize is 512 blocks

The fully-qualified file names for the default keys are:

For Windows
- <pd-home>/log/msg__amj_error1.log
- <pd-home>/log/msg__amj_error2.log
- <pd-home>/log/msg__amj_error3.log

For unix
- <var-dir>/PolicyDirector/log/msg__amj_error1.log
- <var-dir>/PolicyDirector/log/msg__amj_error2.log
- <var-dir>/PolicyDirector/log/msg__amj_error3.log

If either PDJLog.properties or PD.properties is not found, no logging will take place.

This section shows the key/value pairs that may be specified to configure a PDJWarningFileHandler.

The fileName key specifies the fixed part of the name of the file to which the warning output for all the PDJRTE components is written out. The first file will have the base part of this name with the numeral 1 appended. Each subsequent file (controlled by the maxFiles and maxFileSize keys below) will have their respective numbers appended to the base part of the name. For instance, if maxFiles=3 and maxFileSize=10, there will be 3 files of 10 blocks (512 bytes each) written before rollover occurs.

The default fileName is msg__amj_warning.log.
The default maxFiles is 3
The default maxFileSize is 512 blocks

The fully-qualified file names for the default keys are:

For Windows
- <pd-home>/log/msg__amj_warning1.log
- <pd-home>/log/msg__amj_warning2.log
- <pd-home>/log/msg__amj_warning3.log

For unix
- <var-dir>/PolicyDirector/log/msg__amj_warning1.log
- <var-dir>/PolicyDirector/log/msg__amj_warning2.log
- <var-dir>/PolicyDirector/log/msg__amj_warning3.log

Appendix C. Sample PDJLog.properties file
# If either PDJLog.properties or PD.properties is not found, no logging will 
# take place.
#--------------------------------------------------------------------------------

#baseGroup.PDJWarningFileHandler.fileName=
#baseGroup.PDJWarningFileHandler.maxFileSize=
#baseGroup.PDJWarningFileHandler.maxFiles=
baseGroup.PDJWarningFileHandler.isLogging=true
#baseGroup.PDJWarningFileHandler.maxFiles=
#baseGroup.PDJWarningFileHandler.maxFileSize=
#baseGroup.PDJWarningFileHandler.fileName=
#--------------------------------------------------------------------------------
# This section shows the key/value pairs that may be specified to 
# configure a PDJNoticeFileHandler. 
# The fileName key specifies the fixed part of the name of the file to which 
# the notice output for all the PDJRTE components is written out. 
# The first file will have the base part of this name with the numeral 1 
# appended. Each subsequent file (controlled by the maxFiles and 
# maxFileSize keys below) will have their respective numbers appended to the 
# base part of the name. For instance, if maxFiles=3 and maxFileSize=10, 
# there will be 3 files of 10 blocks (512 bytes each) written before rollover 
# occurs.
#
# The default fileName is msg__amj_notice.log. 
# The default maxFiles is 3 
# The default maxFileSize is 512 blocks 
#
# The fully-qualified file names for the default keys are:
#
# For Windows
#  <pd-home>/log/msg__amj_notice1.log
#  <pd-home>/log/msg__amj_notice2.log
#  <pd-home>/log/msg__amj_notice3.log
# For unix
#  <var-dir>/PolicyDirector/log/msg__amj_notice1.log
#  <var-dir>/PolicyDirector/log/msg__amj_notice2.log
#  <var-dir>/PolicyDirector/log/msg__amj_notice3.log
#
# If either PDJLog.properties or PD.properties is not found, no logging will 
# take place. 
#--------------------------------------------------------------------------------

#baseGroup.PDJNoticeFileHandler.fileName=
#baseGroup.PDJNoticeFileHandler.maxFileSize=
#baseGroup.PDJNoticeFileHandler.maxFiles=
baseGroup.PDJNoticeFileHandler.isLogging=false
#baseGroup.PDJNoticeFileHandler.maxFiles=
#baseGroup.PDJNoticeFileHandler.maxFileSize=
#baseGroup.PDJNoticeFileHandler.fileName=
#--------------------------------------------------------------------------------
# This section shows the key/value pairs that may be specified to 
# configure a PDJNoticeVerboseFileHandler. 
# The fileName key specifies the fixed part of the name of the file to which 
# the verbose notice output for all the PDJRTE components is written out. 
# The first file will have the base part of this name with the numeral 1 
# appended. Each subsequent file (controlled by the maxFiles and 
# maxFileSize keys below) will have their respective numbers appended to the 
# base part of the name. For instance, if maxFiles=3 and maxFileSize=10, 
# there will be 3 files of 10 blocks (512 bytes each) written before rollover 
# occurs.
#
# The default fileName is msg__amj_noticeverbose.log. 
# The default maxFiles is 3 
# The default maxFileSize is 512 blocks 
#
# The fully-qualified file names for the default keys are:
#
# For Windows
# <pd-home>/log/msg__amj_noticeverbose1.log
# <pd-home>/log/msg__amj_noticeverbose2.log
# <pd-home>/log/msg__amj_noticeverbose3.log

# For unix
# <var-dir>/PolicyDirector/log/msg__amj_noticeverbose1.log
# <var-dir>/PolicyDirector/log/msg__amj_noticeverbose2.log
# <var-dir>/PolicyDirector/log/msg__amj_noticeverbose3.log

# If either PDJLog.properties or PD.properties is not found, no logging will take place.
#-------------------------------------------------------------

#baseGroup.PDJNoticeVerboseFileHandler.fileName=
#baseGroup.PDJNoticeVerboseFileHandler.maxFileSize=
#baseGroup.PDJNoticeVerboseFileHandler.maxFiles=
baseGroup.PDJNoticeVerboseFileHandler.isLogging=false

#-------------------------------------------------------------
# This section shows the key/value pairs that may be specified to configure a PDJConsoleHandler.
#-------------------------------------------------------------
baseGroup.PDJConsoleHandler.isLogging=false

#-------------------------------------------------------------
# This section shows the key/value pairs that may be specified to configure a PDJTraceConsoleHandler.
#-------------------------------------------------------------
#baseGroup.PDJTraceConsoleHandler.isLogging=

#-------------------------------------------------------------
# This section shows the key/value pairs that may be specified to configure a PDJMessageConsoleHandler.
#-------------------------------------------------------------
#baseGroup.PDJMessageConsoleHandler.isLogging=

#-------------------------------------------------------------
# This section shows the key/value pairs that may be specified to configure a PDJadminTraceLogger.
#-------------------------------------------------------------
#baseGroup.PDJadminTraceLogger.isLogging=

#-------------------------------------------------------------
# This section shows the key/value pairs that may be specified to configure a PDJauditTraceLogger.
#-------------------------------------------------------------
#baseGroup.PDJauditTraceLogger.isLogging=

#-------------------------------------------------------------
# This section shows the key/value pairs that may be specified to configure a PDJasn1TraceLogger.
#-------------------------------------------------------------

Appendix C. Sample PDJLog.properties file 119
This section shows the key/value pairs that may be specified to configure a PDJutilTraceLogger.

This section shows the key/value pairs that may be specified to configure a PDJtsTraceLogger.

This section shows the key/value pairs that may be specified to configure a PDJauthzTraceLogger.

This section shows the key/value pairs that may be specified to configure a PDJsvrsslcfgTraceLogger.

This section shows the key/value pairs that may be specified to configure an PDJAdminAllMaskFilter.

This section shows the key/value pairs that may be specified to configure an PDJAuditAllMaskFilter.

This section shows the key/value pairs that may be specified to configure an PDJutilAllMaskFilter.

This section shows the key/value pairs that may be specified to configure an PDJasn1AllMaskFilter.
This section shows the key/value pairs that may be specified to configure an PDJauthzAllMaskFilter.

```
#baseGroup.PDJauthzAllMaskFilter.mask=
```

This section shows the key/value pairs that may be specified to configure an PDJsvrsslcfgAllMaskFilter.

```
#baseGroup.PDJsvrsslcfgAllMaskFilter.mask=
```

This section shows the key/value pairs that may be specified to configure an PDJutilClassFilter. Classes in PDJLog are treated as subcomponents. Modify the "classes" value to turn on/off the logging of different components. A blank value or absent classes qualifier means all components will be logged.

```
#baseGroup.PDJutilClassFilter.classes=
```

This section shows the key/value pairs that may be specified to configure an PDJauditClassFilter. Classes in PDJLog are treated as subcomponents. Modify the "classes" value to turn on/off the logging of different components. A blank value or absent classes qualifier means all components will be logged.

```
#baseGroup.PDJauditClassFilter.classes=
```

This section shows the key/value pairs that may be specified to configure an PDJadminClassFilter. Classes in PDJLog are treated as subcomponents. Modify the "classes" value to turn on/off the logging of different components. A blank value or absent classes qualifier means all components will be logged.

```
#baseGroup.PDJadminClassFilter.classes=
```

This section shows the key/value pairs that may be specified to configure an PDJLog.properties. Classes in PDJLog are treated as subcomponents. Modify the "classes" value to turn on/off the logging of different components. A blank value or absent classes qualifier means all components will be logged.

```
#baseGroup.PDJLog.properties.classes=
```

This section shows the key/value pairs that may be specified to configure an PDJtsClassFilter. Classes in PDJLog are treated as subcomponents. Modify the "classes" value to turn on/off the logging of different components. Absence of this qualifier means all components will be logged.

```
#baseGroup.PDJtsClassFilter.classes=
```
# configure an PDJauthzClassFilter. Classes in PDJLog are treated as
# subcomponents. Modify the "classes" value to turn on/off the logging
# of different components. Absence of this qualifier means all components
# will be logged.
#--------------------------------------------------------------
#baseGroup.PDJauthzClassFilter.classes=
#--------------------------------------------------------------

# This section shows the key/value pairs that may be specified to
# configure an PDJsyrsslcfgClassFilter. Classes in PDJLog are treated as
# subcomponents. Modify the "classes" value to turn on/off the logging
# of different components. Absence of this qualifier means all components
# will be logged.
#--------------------------------------------------------------

#baseGroup.PDJsvrsslcfgClassFilter.classes=
Appendix D. User registry differences

The following user registry differences are known to exist in this version of IBM Tivoli Access Manager (Tivoli Access Manager).

1. When Tivoli Access Manager is using either Microsoft Active Directory or a Lotus Domino server as its user registry, only a single domain is supported. Use an LDAP user registry if you wish to take advantage of the multi-domain support in Tivoli Access Manager.

2. Tivoli Access Manager does not support cross domain group membership or universal groups when using Microsoft Active Directory as its user registry. Importing such groups into Tivoli Access Manager is not supported.

3. When the Tivoli Access Manager policy server is using either Microsoft Active Directory or a Lotus Domino server as its user registry, existing Tivoli SecureWay Policy Director, Version 3.8 clients are not able to connect to the policy server. Either use a different user registry or upgrade the clients to Tivoli Access Manager.

4. Users created in a Lotus Domino server or Microsoft Active Directory user registry are automatically given the capability to own single signon credentials and this capability can not be removed. When using an LDAP user registry, this capability must be explicitly granted to a user and subsequently can be removed.

5. Leading and trailing blanks in user names and group names are ignored when using LDAP or Microsoft Active Directory as the user registry in an Tivoli Access Manager secure domain. However, when using a Lotus Domino server as a user registry, leading and trailing blanks are significant. To ensure that processing is consistent regardless of what user registry is being used, define users and groups in the user registry without leading or trailing blanks in their names.

6. The forward slash character (/) should be avoided in user and group names defined using distinguished name strings. The forward slash character is treated differently in different user registries:

**Lotus Domino server**

Users and groups can not be created with names using a distinguished name string containing a forward slash character. To avoid the problem, either do not use a forward slash character or define the user without using the distinguished name designation:

```
padmin user create myuser username/locinfo test testpwd
```

instead of using this one:

```
padmin user create myuser cn=username/o=locinfo test testpwd
```

**Microsoft Active Directory**

Users and groups can be created with names using a distinguished name string containing a forward slash character. However, subsequent operations on the object might fail as some Active Directory functions interpret the forward slash character as a separator between the object name and the host name. To avoid the problem, do not use a forward slash character to define the user.

7. When using a multi-domain Microsoft Active Directory user registry, multiple users and groups can be defined with the same short name as long as they
reside in different domains. However, the full name of the user or group, including the domain suffix, must always be specified to Tivoli Access Manager.

8. When using iPlanet Version 5.0 as the user registry, a user that is created, added to a group, and then deleted from the user registry retains its group membership. If a user with the same name is created at some later time, the new user automatically inherits the old group membership and might be given inappropriate permissions. It is strongly recommended that the user be removed from all groups before the user is deleted. This problem does not occur when using the other supported user registries.

9. Attempting to add a single duplicate user to a group does not produce an error when an LDAP user registry is being used. However, an error is properly reflected when using Lotus Domino server or Microsoft Active Directory.

10. The Tivoli Access Manager authorization API provides a credentials attribute entitlements service. This service is used to retrieve user attributes from a user registry. When this service is used with an LDAP user registry, the retrieved attributes can be either string or binary data. However, when this service is used with a Microsoft Active Directory or Lotus Domino user registry, the retrieved attributes can be either string, binary or integer data.

11. The maximum lengths of various names associated with Tivoli Access Manager vary depending on the user registry being used. See Table 35 for a comparison of the maximum lengths allowed and the recommended maximum length to use to ensure compatibility with all the user registries supported by Tivoli Access Manager.

Table 35. Maximum lengths for names based on user registry

<table>
<thead>
<tr>
<th>Maximum length of:</th>
<th>LDAP</th>
<th>Microsoft Active Directory</th>
<th>Lotus Domino server</th>
<th>Recommended maximum value</th>
</tr>
</thead>
<tbody>
<tr>
<td>First name (LDAP CN)</td>
<td>256</td>
<td>64</td>
<td>960</td>
<td>64</td>
</tr>
<tr>
<td>Middle name</td>
<td>128</td>
<td>64</td>
<td>65535</td>
<td>64</td>
</tr>
<tr>
<td>Last name (surname)</td>
<td>128</td>
<td>64</td>
<td>960</td>
<td>64</td>
</tr>
<tr>
<td>Registry UID (LDAP DN)</td>
<td>1024</td>
<td>2048</td>
<td>255</td>
<td>This value is user registry-specific and must be changed when changing user registries.</td>
</tr>
<tr>
<td>Tivoli Access Manager user identity</td>
<td>256</td>
<td>2048 - 1 - length_of_domain_name</td>
<td>200 - 4 - length_of_domain_name</td>
<td>This value is user registry-specific and must be changed when changing user registries.</td>
</tr>
<tr>
<td>User password</td>
<td>unlimited</td>
<td>256</td>
<td>unlimited</td>
<td>256</td>
</tr>
<tr>
<td>User description</td>
<td>1024</td>
<td>1024</td>
<td>unlimited</td>
<td>1024</td>
</tr>
<tr>
<td>Group name</td>
<td>256</td>
<td>1024</td>
<td>1024</td>
<td>256</td>
</tr>
<tr>
<td>Group description</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
</tr>
</tbody>
</table>
Even though some names can be of unlimited length, excessive lengths can result in policy that is difficult to manage and might result in poor system performance. Choose maximum values that are logical for your environment.

### Table 35. Maximum lengths for names based on user registry (continued)

<table>
<thead>
<tr>
<th>Maximum length of:</th>
<th>LDAP</th>
<th>Microsoft Active Directory</th>
<th>Lotus Domino server</th>
<th>Recommended maximum value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single signon resource name</td>
<td>240</td>
<td>256</td>
<td>256</td>
<td>240</td>
</tr>
<tr>
<td>Single signon resource description</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
</tr>
<tr>
<td>Single signon user ID</td>
<td>240</td>
<td>256</td>
<td>256</td>
<td>240</td>
</tr>
<tr>
<td>Single signon password</td>
<td>unlimited</td>
<td>256</td>
<td>unlimited</td>
<td>256</td>
</tr>
<tr>
<td>Single signon group name</td>
<td>240</td>
<td>256</td>
<td>256</td>
<td>240</td>
</tr>
<tr>
<td>Single signon group description</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
</tr>
<tr>
<td>Action name</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Action description, action type</td>
<td>unlimited</td>
<td>unlimited</td>
<td>unlimited</td>
<td></td>
</tr>
<tr>
<td>Object name, object space name, ACL name, POP name</td>
<td>unlimited</td>
<td>unlimited</td>
<td>unlimited</td>
<td></td>
</tr>
<tr>
<td>Object description, object space description, ACL</td>
<td>unlimited</td>
<td>unlimited</td>
<td>unlimited</td>
<td></td>
</tr>
<tr>
<td>description, POP description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix E. 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
500 Columbus Avenue
Thornwood, NY 10594
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, 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
2Z4A/101
11400 Burnet Road
Austin, TX 78758
USA

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 document 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 measurement 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.

All statements regarding IBM’s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM’s application programming interfaces.
Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

**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  
Notes  
Lotus Notes  
SecureWay  
Tivoli  
Tivoli logo  

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.
## Index

### A
- API (C) collecting defect data 105
- AutoTrace
  - AutoTrace file descriptions 61
  - building 60
  - config file 61
  - config file description 63
  - create trace ID file 71
  - exclude file description 65
  - initializing atctl.exe 65
  - initializing atserv.exe 65
  - output format and example overview 67
  - product file 61
  - product file description 63
  - useful commands 69
  - using 66
  - aznapi-configuration stanza 83

### C
- commands
  - pdacld_dump 9
  - pdbackup 8
  - pdversion 5
- customer support
  - collecting data 105
  - collecting defect data for pdadmin and C APIs 105
  - contacting 105

### D
- deprecated commands
  - pdinfo 8

### E
- environment variables
  - PD_HOME 5
  - PD_SVC_ROUTING_FILE 28, 45
  - SVC_ERROR 33
  - SVC_FATAL 33
  - SVC_NOTICE 33
  - SVC_NOTICE_VERBOSE 33
  - SVC_WARNING 33
- event logging
  - statistics 83

### G
- GSKit
determining version 6

### I
- IBM Global Security Kit
determining version 6

### IBM Tivoli Directory Client
  - determining version 6
- IBM Tivoli Directory Server
  - determining version 6
  - installation
    - configuration log files 17
  - installing
    - installation log files 17
    - installation log files from easy install 16

### L
- LDAP-based user registry
  - determining version 6
  - logcfg 83

### M
- message logging
  - log entry format 25

### P
- pdacld_dump
  - syntax 13
- pdacld_dump command 9
- pdadmin
  - collecting defect data 105
  - pdbackup command 8
  - pdinfo command (deprecated) 8
  - pdversion command 5
- pdweb.authn statistics 78
- pdweb.authz statistics 78
- pdweb.docache statistics 81
- pdweb.http statistics 79
- pdweb.https statistics 79
- pdweb.jct.# statistics 83
- pdweb.jmt statistics 80
- pdweb.sescache statistics 80
- pdweb.threads statistics 80
- problem determination
  - overview 1
    - problem prevention 1

### R
- related publications x

### S
- statistics
  - activity types 78
  - components 78
  - disable (stats off) 77
  - display (stats get) 77
  - enable (stats on) 75
  - enable using event logging 83
  - list (stats list) 78
  - logcfg parameter 83

© Copyright IBM Corp. 2002, 2003
statistics (continued)
  pdweb.authn  78
  pdweb.authz  78
  pdweb.doccache  81
  pdweb.http  79
  pdweb.https  79
  pdweb.jct.#  83
  pdweb.jmt  80
  pdweb.sescache  80
  pdweb.threads  80
  reset (stats reset)  77
  show (stats show)  77
  stats command syntax  75
  stats commands  75
  stats parameter  83
  statistics (WebSEAL)  75
  stats  83
  stats, commands  75

T
  trace
    capturing actions  48
    syntax  48
    trace entry format  57
    trace log file  51
  trace, pdadmin trace  43
  troubleshooting
    Active Directory  89
    Domino Server  90
    GSKit  87
    IBM Tivoli Directory Client  88
    junctioned servers  92
    LDAP server  87
    Tivoli Access Manager authorization server  91
    Tivoli Access Manager Plug-in for Web Servers  92
    Tivoli Access Manager policy server  91
    Tivoli Access Manager runtime  91
    Tivoli Access Manager servers  90
    Tivoli Access Manager WebSEAL  91
    verifying component functionality  86
    verifying software installation  85

U
  user registry
    differences  123
    maximum values  124, 125

V
  version
    IBM Global Security Kit  6
    IBM Tivoli Directory Client  6
    IBM Tivoli Directory Server  6
    LDAP-based user registry  6

W
  WebSEAL
    statistics  75