Note
Before using this information and the product it supports, read the information in Appendix E, “Notices,” on page 357.
# Contents

## Preface

Who should read this book .......................................................... xi
What this book contains ............................................................. xi
Publications ................................................................................. xiii
  - Release information ............................................................... xiii
  - Base information ................................................................. xiii
  - Web security information ...................................................... xiii
  - Developer references ............................................................ xiv
  - Technical supplements .......................................................... xv
  - Related publications ............................................................. xv
  - Accessing publications online ............................................... xviii
Accessibility ................................................................................ xviii
Contacting software support ....................................................... xviii
Conventions used in this book ...................................................... xix
  - Typeface conventions ............................................................ xix
  - Operating system differences ................................................ xix

## Chapter 1. IBM Tivoli Access Manager overview

Core technologies .......................................................................... 1
  - Authentication ......................................................................... 2
  - Authorization .......................................................................... 2
  - Quality of protection .................................................................. 2
  - Scalability ................................................................................ 3
  - Accountability .......................................................................... 3
  - Centralized management .......................................................... 3
Security policy overview ............................................................... 4
Authorization API standard .......................................................... 5
Authorization: conceptual model .................................................. 5
  - The benefits of a standard authorization service ....................... 7
  - Tivoli Access Manager authorization service overview ............ 7
The Tivoli Access Manager authorization service .......................... 8
  - Components ............................................................................ 8
  - Authorization service interfaces .............................................. 9
  - Replication for scalability and performance .............................. 9
Implementing a network security policy ......................................... 11
  - Defining and applying security policy ...................................... 11
  - The authorization process: step-by-step ................................... 12
The Tivoli Access Manager authorization API ................................ 13
  - Using the authorization API: two examples .............................. 14
  - Authorization API: remote cache mode ................................... 15
  - Authorization API: local cache mode ........................................ 16
External authorization capability .................................................. 17
  - Extending the authorization service ........................................ 17
  - Imposing conditions on resource requests ............................... 18
  - The authorization evaluation process ...................................... 18
  - Implementing an external authorization service ....................... 20
  - Deployment strategies ............................................................ 20

## Chapter 2. Web Portal Manager

Types of administration ............................................................... 23
Delegate administration functions .............................................. 24
  - Self-care .................................................................................. 24
  - Self-registration ................................................................. 24
Web Portal Manager common tasks .......................................... 25
  - Start Web Portal Manager administration ............................... 25
Chapter 3. Tivoli Access Manager administration ........................................... 33
Domains ........................................................................................................... 33
Protected object space ..................................................................................... 34
Users and groups .............................................................................................. 36
Security policy .................................................................................................... 36
ACL policies ....................................................................................................... 38
  Using ACL policies with the authorization service ........................................ 38
  Evaluating an ACL .......................................................................................... 38
Protected object policies .................................................................................... 39
Authorization rules .............................................................................................. 40
  How authorization rules differ from ACLs and POPs .................................... 40
  When to use authorization rules ..................................................................... 40
Guidelines for a secure object space ................................................................. 41

Chapter 4. Default security policy ...................................................................... 43
Default administration users and groups ......................................................... 43
  group iv-admin ................................................................................................ 43
  user sec_master .............................................................................................. 43
  group ivmgd-servers ..................................................................................... 43
  Administration users ...................................................................................... 43
Example administration ACL policies .............................................................. 44
Defining and applying security policy .............................................................. 44
  Access control list .......................................................................................... 45
  Default administration ACL policies ............................................................ 45
Management permissions .................................................................................. 47
Sparse security policy model ............................................................................. 54
Protected object policies .................................................................................... 58
Authorization rules .............................................................................................. 59

Chapter 5. Domain management. ........................................................................ 61
Create a domain. ................................................................................................. 61
  Web Portal Manager pdadmin ...................................................................... 61
List domains ....................................................................................................... 62
  Web Portal Manager pdadmin ...................................................................... 62
Delete a domain. ................................................................................................. 62
  Web Portal Manager pdadmin ...................................................................... 62
Modify a domain ................................................................................................. 63
  Web Portal Manager pdadmin ...................................................................... 63

Chapter 6. Object space management .................................................................. 65
Create an object space ....................................................................................... 65
  Web Portal Manager pdadmin ...................................................................... 65
Contents

padmin ......................................................... 66
Create an object ........................................... 66
  Web Portal Manager ..................................... 66
  pdadmin .................................................. 67
Browse the object space .................................. 67
  Web Portal Manager ..................................... 67
  pdadmin .................................................. 68
Delete an object space .................................... 68
  Web Portal Manager ..................................... 68
  pdadmin .................................................. 68

Chapter 7. Protected object management .................... 69
Create an object ........................................... 69
  Web Portal Manager ..................................... 69
  pdadmin .................................................. 69
List objects ................................................ 70
  Web Portal Manager ..................................... 70
  pdadmin .................................................. 70
Delete an object .......................................... 70
  Web Portal Manager ..................................... 70
  pdadmin .................................................. 70

Chapter 8. Access control list management .................. 73
ACL policies ................................................ 73
ACL entries ................................................. 73
  Type attribute .......................................... 74
  ID attribute ............................................. 74
  Permissions (actions) attribute ......................... 75
  Default Tivoli Access Manager permissions (actions) ............................................. 75
Managing an access control list ......................... 76
  Create an ACL ........................................... 76
  List ACLs ................................................ 76
  Delete an ACL ........................................... 76
  Modify an ACL ........................................... 77
Applying an ACL attribute to a protected object ........... 78
  Attach an ACL to an object ............................ 78
  Find where an ACL is attached ......................... 79
  Detaching an ACL ....................................... 79
Example ACL entries ........................................ 80
ACL policies and the protected object space ............... 80
  Root (/) container object ................................................ 80
  The traverse permission ..................................... 80
Object and object space permissions ...................... 81
Action group management ................................ 81
  Action groups .......................................... 82
  Create a new action group ............................. 82
  List action groups ..................................... 83
  Delete an action group .................................. 83
  Create new actions in an action group .................. 84
  Custom actions ......................................... 84
  Entering custom actions into ACL entries ............... 85

Chapter 9. Protected object policy management ............... 87
Managing protected object policies ....................... 88
  Create a POP .......................................... 88
  List POPs ................................................ 90
  Delete a POP ............................................ 90
  Modify a POP ............................................ 91
Applying POP attributes to protected objects .............. 91
  Attach a POP to an object ............................. 92
Find where a POP is attached ............................................. 92
Detach a POP from a protected object ................................. 92
Configuring the POP attributes ........................................... 93
Warning mode attribute .................................................... 93
Audit level attribute ....................................................... 93
Time-of-day attribute ..................................................... 94
Network-based authorization POP policy ............................... 95
Specifying IP addresses and ranges .................................... 95
Disabling network-based authorization by IP address ............. 96
Network-based authorization algorithm ............................... 96
Network-based authorization notes and limitations .................. 96
Authentication strength POP policy (step-up) ......................... 97
Configuring levels for step-up authentication ....................... 97
Applying step-up authentication policy ............................... 98
Distinguishing step-up from multi-factor authentication ........... 98
Quality of protection POP policy ....................................... 99

Chapter 10. Authorization rules management. ......................... 101
Authorization rules overview ............................................. 101
Access decision information ............................................. 101
Sources for retrieving ADI ............................................... 101
Volatile versus non-violatilte data ................................... 103
Authorization rule language ............................................. 104
ADI XML document model .............................................. 104
XML access decision information .................................... 106
Defining an XML namespace .......................................... 108
Authorization rules evaluator ........................................... 110
Format and constraints of rules ....................................... 111
Example rules ............................................................ 113
Methods of providing ADI to the rules evaluator .................... 116
Reason codes for rule failures ........................................ 117
Configuration file and initialization attributes ....................... 118
resource-manager-provided-adi ...................................... 118
dynamic-adi-entitlement-services .................................. 118
input-adi-xml-prolog and xsl-stylesheet-prolog .................... 119
[xmladi-attribute-definitions] ....................................... 119
Managing an authorization rule ....................................... 120
Create an authorization rule .......................................... 120
List authorization rules ............................................... 121
Delete an authorization rule ......................................... 121
Modify an authorization rule ....................................... 122
Attach an authorization rule to a protected object ............... 122
Finding objects having authorization rules attached ............... 123
Detaching an authorization rule ..................................... 124

Chapter 11. User and group management ............................... 125
Search for users ................................................................ 125
Web Portal Manager ....................................................... 125
padmin ................................................................. 126
Create a user .................................................................. 126
Web Portal Manager ....................................................... 126
padmin ................................................................. 127
Import a user .................................................................. 127
Web Portal Manager ....................................................... 127
padmin ................................................................. 128
Set global user policies ............................................... 128
Web Portal Manager ....................................................... 128
padmin ................................................................. 129
Change user password ................................................... 130
Web Portal Manager ....................................................... 130
# Delegated administration

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search for groups</td>
<td>130</td>
</tr>
<tr>
<td>- Web Portal Manager</td>
<td>130</td>
</tr>
<tr>
<td>- pdadmin</td>
<td>131</td>
</tr>
<tr>
<td>Create a group</td>
<td>131</td>
</tr>
<tr>
<td>- Web Portal Manager</td>
<td>131</td>
</tr>
<tr>
<td>- pdadmin</td>
<td>131</td>
</tr>
<tr>
<td>Import a group</td>
<td>132</td>
</tr>
<tr>
<td>- Web Portal Manager</td>
<td>132</td>
</tr>
<tr>
<td>- pdadmin</td>
<td>132</td>
</tr>
</tbody>
</table>

## Chapter 12. Certificate and password management

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial configuration</td>
<td>134</td>
</tr>
<tr>
<td>Key file and stash file renewal information</td>
<td>135</td>
</tr>
<tr>
<td>Trust determination</td>
<td>136</td>
</tr>
<tr>
<td>Server certificate revocation</td>
<td>136</td>
</tr>
<tr>
<td>Additional key and stash file considerations</td>
<td>137</td>
</tr>
</tbody>
</table>

## Chapter 13. Server management

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tivoli Access Manager servers</td>
<td>139</td>
</tr>
<tr>
<td>- Proxy server</td>
<td>140</td>
</tr>
<tr>
<td>- Server dependencies</td>
<td>141</td>
</tr>
<tr>
<td>Tivoli Access Manager utilities</td>
<td>142</td>
</tr>
<tr>
<td>Tivoli Access Manager servers tasks</td>
<td>142</td>
</tr>
<tr>
<td>- Start and stop servers on UNIX systems</td>
<td>142</td>
</tr>
<tr>
<td>- Start and stop servers on Windows systems</td>
<td>143</td>
</tr>
<tr>
<td>Server configuration file tasks</td>
<td>144</td>
</tr>
<tr>
<td>- Change configuration settings</td>
<td>144</td>
</tr>
<tr>
<td>- Automate server startup at boot time</td>
<td>145</td>
</tr>
<tr>
<td>Policy server administration tasks</td>
<td>145</td>
</tr>
<tr>
<td>- Replicate the authorization database</td>
<td>145</td>
</tr>
<tr>
<td>- Set the number of update-notifier threads</td>
<td>147</td>
</tr>
<tr>
<td>- Set the notification delay time</td>
<td>147</td>
</tr>
</tbody>
</table>

## Chapter 14. High availability of policy server

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data integrity</td>
<td>149</td>
</tr>
<tr>
<td>Primary and replica LDAP servers</td>
<td>149</td>
</tr>
<tr>
<td>Active and passive policy servers</td>
<td>149</td>
</tr>
<tr>
<td>High availability management</td>
<td>150</td>
</tr>
<tr>
<td>- Verify the high-availability policy servers setup</td>
<td>150</td>
</tr>
<tr>
<td>- Review log files</td>
<td>151</td>
</tr>
</tbody>
</table>

## Chapter 15. Multiple-tenancy policy server

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of delegated administration</td>
<td>155</td>
</tr>
<tr>
<td>Delegated role administration</td>
<td>157</td>
</tr>
<tr>
<td>Delegated object space management</td>
<td>158</td>
</tr>
<tr>
<td>- Structuring the object space for management delegation</td>
<td>159</td>
</tr>
<tr>
<td>- Default administration users and groups</td>
<td>159</td>
</tr>
<tr>
<td>- Example: Management delegation</td>
<td>159</td>
</tr>
<tr>
<td>Delegated user and group management</td>
<td>160</td>
</tr>
<tr>
<td>- Create group container objects</td>
<td>161</td>
</tr>
<tr>
<td>- Create groups</td>
<td>162</td>
</tr>
<tr>
<td>- ACL policies affecting group management</td>
<td>164</td>
</tr>
<tr>
<td>- ACL policies affecting user management</td>
<td>165</td>
</tr>
<tr>
<td>Delegated administration security policy</td>
<td>166</td>
</tr>
</tbody>
</table>

## Chapter 17. Log and routing files

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
</table>

[Contents vii]
Chapter 18. XML output for logging and auditing logs .......................................................... 179
DTD intermediate format ........................................................................................................ 179
Data blocks and output fields .................................................................................................. 179
Action codes for management commands .............................................................................. 191
Data output for authentication errors .................................................................................... 196

Chapter 19. Capturing logging and auditing events ................................................................. 197
Log agents ................................................................................................................................ 197
Categories of events ................................................................................................................ 197
  Program-specific event category .......................................................................................... 197
  Event pool hierarchy ............................................................................................................. 198
Configuration parameters for EventPool ................................................................................. 198
  EventPool .............................................................................................................................. 201
  queue_size .......................................................................................................................... 201
  hi-water ............................................................................................................................... 201
  flush_interval ....................................................................................................................... 202
Configuration parameters for console log agent ....................................................................... 202
  stdout ................................................................................................................................. 202
  stderr ................................................................................................................................. 203
Configuration parameters for a file log agent ......................................................................... 203
  path ...................................................................................................................................... 203
  logcfg .................................................................................................................................. 203
  rollover_size ...................................................................................................................... 204
  buffer_size ......................................................................................................................... 204
  queue_size .......................................................................................................................... 205
  hi_water .............................................................................................................................. 206
  flush_interval ..................................................................................................................... 206
  mode ..................................................................................................................................... 207
Configuration parameters for pipe log agent ......................................................................... 207
  path ...................................................................................................................................... 207
  queue_size .......................................................................................................................... 207
  hi_water .............................................................................................................................. 208
  flush_interval ..................................................................................................................... 208
Configuration parameters for remote log agent ..................................................................... 208
  buffer_size .......................................................................................................................... 209
  flush_interval ..................................................................................................................... 209
  queue_size and hi_water ..................................................................................................... 209
  compress ............................................................................................................................. 209
  error .................................................................................................................................... 210
  path ...................................................................................................................................... 210
  rebind_retry ....................................................................................................................... 210
  server ................................................................................................................................... 210
  port ....................................................................................................................................... 210
  dn .......................................................................................................................................... 210
Event logging tasks .................................................................................................................. 211
  Send event records to a log file ......................................................................................... 211
  Pipe event records to another program ............................................................................. 213
Send event records to a console ........................................... 214
Send event records to a remote server ................................. 215
Monitor log queue performance ........................................ 217

Chapter 20. Logging of legacy auditing events ...................... 219
Overview of auditing ...................................................... 219
Audit trail file contents .................................................. 220
   Examples of authorization audit records .......................... 220
   Examples of authentication audit records ....................... 221
   Examples of management audit records ......................... 223
Legacy configuration file stanza entries for auditing ............... 225
   logaudit .............................................................. 225
   auditcfg .......................................................... 225
   auditlog .......................................................... 226
   logsize ........................................................... 227
   logflush .......................................................... 227
   audit-attribute .................................................... 227
Logging and auditing tasks .............................................. 228
   Enable auditing .................................................... 228
   Specify which ADI attributes to audit ......................... 229
   Maintain audit log files .......................................... 229

Appendix A. Server configuration file reference ..................... 231
Configuration files ...................................................... 231
   activeidir.conf .................................................. 232
   activeidir_ldap.conf ............................................ 232
   domino.conf ..................................................... 233
   ivacld.conf ...................................................... 233
   ivmgrd.conf ...................................................... 234
   ldap.conf ......................................................... 234
   pd.conf .......................................................... 235
   pdmgrproxyd.conf ............................................... 235
   pdwpml.conf ...................................................... 236
Resource manager configuration files ................................. 237
Guidelines for configuring stanzas .................................... 238
   General guidelines ............................................... 238
   Default values ................................................... 238
   Strings ............................................................ 239
   Defined strings ................................................... 239
   File names ........................................................ 239
   Integers ........................................................... 240
   Boolean values ..................................................... 240
Stanzas ........................................................................ 241
   [authentication-mechanisms] stanza ................................ 242
   [aznapi-admin-services] stanza .................................. 245
   [aznapi-configuration] stanza .................................... 247
   [aznapi-cred-modification-services] stanza .................... 258
   [aznapi-entitlement-services] stanza ........................... 260
   [aznapi-external-authn-services] stanza ......................... 262
   [aznapi-pac-services] stanza ..................................... 264
   [configuration-database] stanza .................................. 266
   [delegated-admin] stanza ......................................... 267
   [domains] and [domain=domain_name] stanzas .................. 268
   [ivacld] stanza ..................................................... 270
   [ivmgrd] stanza ..................................................... 273
   [ldap] stanza ....................................................... 278
   [ldap] stanza for ldap.conf ...................................... 287
   [manager] stanza .................................................. 292
   [meta-info] stanza ................................................ 294
   [pdmgrproxyd] stanza .............................................. 295
Preface

The IBM Tivoli Access Manager Base Administration Guide provides a comprehensive set of procedures and reference information for managing IBM Tivoli Access Manager (Tivoli Access Manager) servers and resources. This guide also provides you with valuable background and concept information for the wide range of Tivoli Access Manager functionality.

Who should read this book

This guide is for system administrators responsible for the deployment and administration of base Tivoli Access Manager software.

Readers should be familiar with the following:
• Microsoft® Windows® and UNIX® operating systems
• Database architecture and concepts
• Security management
• Internet protocols, including HTTP and TCP/IP
• Lightweight Directory Access Protocol (LDAP) and directory services
• Authentication and authorization
• Tivoli Access Manager security model and its capabilities

You also should be familiar with SSL protocol, key exchange (public and private), digital signatures, cryptographic algorithms, and certificate authorities.

What this book contains

This guide contains the following sections:
• Chapter 1, “IBM Tivoli Access Manager overview,” on page 1
  Provides an overview of IBM Tivoli Access Manager.
• Chapter 2, “Web Portal Manager,” on page 23
  Describes the two types of Web Portal Manager administration functions.
• Chapter 3, “Tivoli Access Manager administration,” on page 33
  Describes the administration of Tivoli Access Manager.
• Chapter 4, “Default security policy,” on page 43
  Describes the default security policy provided with Tivoli Access Manager.
• Chapter 5, “Domain management,” on page 61
  Describes how the administrator in the management domain can create additional domains, and describes which administrative tasks the domain administrator can perform within their own domain.
• Chapter 6, “Object space management,” on page 65
  Describes the tasks associated with managing object spaces.
• Chapter 7, “Protected object management,” on page 69
  Describes the tasks associated with managing objects.
• Chapter 8, “Access control list management,” on page 73
  Describes the tasks associated with managing access control lists (ACLs).
• Chapter 9, “Protected object policy management,” on page 87
Describes the tasks associated with managing protected object policies (POPs).

- **Chapter 10, “Authorization rules management,” on page 101**
  Provides detailed information about authorization rules (AuthzRules) that are used to make access decisions.

- **Chapter 11, “User and group management,” on page 125**
  Describes the tasks associated with managing users and groups.

- **Chapter 12, “Certificate and password management,” on page 133**
  Describes the tasks associated with managing certificates and passwords.

- **Chapter 13, “Server management,” on page 139**
  Describes the tasks associated with managing the servers associated with Tivoli Access Manager.

- **Chapter 14, “High availability of policy server,” on page 149**
  Describes the tasks associated with creating a high availability environment through the use of policy proxy servers, multiple policy servers, and multiple authorization servers.

- **Chapter 15, “Multiple-tenancy policy server,” on page 153**
  Describes multiple tenancy servers and how to set up a multiple-tenancy Tivoli Access Manager policy server.

- **Chapter 16, “Delegated administration,” on page 155**
  Describes the tasks associated with delegating administration.

- **Chapter 17, “Log and routing files,” on page 169**
  Describes the tasks associated with managing routing files, a common directory for maintaining log files, and how to understand and view Extensible Markup Language (XML) log file output for message and trace log files.

- **Chapter 18, “XML output for logging and auditing logs,” on page 179**
  Describes information about the intermediate DTD format, the XML output from the auditing log files, and the XML tags used for Tivoli Access Manager event logging and auditing.

- **Chapter 19, “Capturing logging and auditing events,” on page 197**
  Describes the tasks associated with capturing logging and auditing events.

- **Chapter 20, “Logging of legacy auditing events,” on page 219**
  Describes the tasks associated with managing legacy logging and auditing.

- **Appendix A, “Server configuration file reference,” on page 231**
  Provides detailed information on the syntax of the configuration files used by Tivoli Access Manager servers.

- **Appendix B, “User registry differences,” on page 325**
  Describes the differences known to exist in the current Tivoli Access Manager user registries: LDAP, Microsoft Active Directory, and IBM Lotus Domino.

- **Appendix C, “Administration command line and Web Portal Manager equivalents,” on page 329**
  Maps the Tivoli Access Manager pdadmin command-line interface functions to the Web Portal Manager graphical user interface functions.

- **Appendix D, “Managing user registries,” on page 339**
  Describes a subset of user registry tasks that are specific to the installation of Tivoli Access Manager.
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”
- “Developer references” on page xiv
- “Technical supplements” on page xv

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 IBM 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 [http://www.ibm.com/software/tivoli/library/](http://www.ibm.com/software/tivoli/library/)

**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<sup>®</sup> 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 z/OS/390<sup>®</sup> LDAP servers as the user registry for Tivoli Access Manager. Note that z/OS and OS/390 are not provided with IBM Tivoli Access Manager Version 5.1.

Additional information about DB2 can be found at:

http://www.ibm.com/software/data/db2/

**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 IBM Directory Server Web Administration Tool 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<sup>®</sup>, Version 5.2, and IBM WebSphere<sup>®</sup> 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, 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](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. IBM Tivoli Access Manager overview

IBM Tivoli Access Manager (Tivoli Access Manager) is an authentication and authorization solution for corporate Web, client/server, and existing applications. Tivoli Access Manager allows you to control user access to protected information and resources. By providing a centralized, flexible, and scalable access control solution, Tivoli Access Manager allows you to build secure and easy-to-manage network-based applications and e-business infrastructure.

Tivoli Access Manager supports authentication, authorization, data security, and resource management capabilities. You use Tivoli Access Manager in conjunction with standard Internet-based applications to build highly secure and well-managed intranets.

Tivoli Access Manager provides:

**Authentication framework**
- Tivoli Access Manager provides a wide range of built-in authenticators and supports external authenticators.

**Authorization framework**
- The Tivoli Access Manager authorization service, accessed through a standard authorization application programming interface (authorization API), provides permit and deny decisions on access requests for native Tivoli Access Manager servers and other applications.

- The authorization service, together with resource managers, provides a standard authorization mechanism for business network systems.

Tivoli Access Manager can be integrated into existing legacy and emerging infrastructures to provide secure, centralized policy management capability.

Some existing resource managers include:

**IBM Tivoli Access Manager WebSEAL**
- Manages and protects Web-based information and resources. WebSEAL is included with Tivoli Access Manager.

**IBM Tivoli Access Manager for Business Integration**
- Provides a security solution for IBM MQSeries and IBM WebSphere MQ messages.

**IBM Tivoli Access Manager for Operating Systems**
- Provides a layer of authorization policy enforcement on UNIX systems in addition to that provided by the native operating system.

Existing applications can take advantage of the Tivoli Access Manager authorization service as well as provide a common security policy for the entire enterprise.
Core technologies

The Tivoli Access Manager network security management solution provides and supports the following core technologies:

- Authentication
- Authorization
- Quality of protection
- Scalability
- Accountability
- Centralized management

Authentication

Authentication is the first step a user must take when making a request for a resource that is protected by Tivoli Access Manager. During authentication, a user’s identity is validated. The authentication process is usually dependent on the specific requirements of the service-providing application. Tivoli Access Manager allows a highly flexible approach to authentication through the use of the authorization API.

Tivoli Access Manager provides built-in support of user name and password authentication through the authorization API. Applications can build any custom authentication mechanism that uses the authorization API.

Authorization

Authorization enforces the security policy by determining what objects a user can access and what actions a user can take on those objects and then granting appropriate access to the user. Tivoli Access Manager handles authorization through the use of the following:

- Tivoli Access Manager authorization service
- Access control lists (ACLs), protected object policies (POPs), and authorization rules for fine-grained access control
- Standards-based authorization API, using the aznAPI for C language applications, and the Java Authentication and Authorization Service (JAAS) for Java language applications
- External authorization service capability

Quality of protection

Quality of protection is the degree to which Tivoli Access Manager protects any information transmitted between client and server. The quality of data protection is determined by the combined effect of encryption standards and modification-detection algorithms. The resource manager is responsible for ensuring that the quality of data protection is enforced.

Quality of protection levels include:

- Standard Transmission Control Protocol (TCP) communication (no protection)
- Data integrity – protects messages (data stream) from being modified during network communication
- Data privacy – protects messages from being modified or inspected during network communication
Tivoli Access Manager supports the data integrity and data privacy provided by the Secure Sockets Layer (SSL) communication protocol. Encryption ciphers supported include the following:

- 40-bit RC2
- 128-bit RC2
- 40-bit RC4
- 128-bit RC4
- 40-Data Encryption Standard (DES)
- 56-bit DES
- 168-bit triple DES

**Scalability**

Scalability is the ability to respond to increasing numbers of users who access resources in the domain. Tivoli Access Manager uses the following techniques to provide scalability:

- Replication of services
  - Authentication services
  - Authorization services
  - Security policies
  - Data encryption services
  - Auditing services
- Front-end replicated servers
  - Mirrored resources for high availability
  - Load balancing client requests
- Back-end replicated servers
  - Back-end servers can be Tivoli Access Manager WebSEAL, Tivoli Access Manager for Operating Systems, Tivoli Access Manager for Business Integration, or other application servers
  - Mirrored resources (unified object space) for high availability
  - Additional content and resources
  - Load balancing of incoming requests
- Optimized performance by allowing the off-loading of authentication and authorization services to separate servers
- Scaled deployment of services without increasing management overhead

**Accountability**

Tivoli Access Manager provides a number of logging and auditing capabilities. Log files capture any error and warning messages generated by Tivoli Access Manager servers. Audit trail files monitor Tivoli Access Manager server activity.

**Centralized management**

Three methods are provided for managing security policy and the Tivoli Access Manager servers:

- **padmin** command line utility
- Web Portal Manager graphical user interface (GUI)
- Administration API
You can accomplish most tasks using any of these methods. However some tasks can not be performed using the Web Portal Manager.

**padmin command line utility**
The `padmin` command line utility is used for Tivoli Access Manager administration. This utility provides a command line interface for managing users, groups, roles, permissions, policies, domains, and servers, as well as for performing other tasks. The command can be used in scripts or batch files to automate processing.

The `padmin` command line utility is installed as part of the Tivoli Access Manager runtime package.

For specific task information, see the task-specific chapters in this guide. For detailed syntax information on the `padmin` command itself, see the *IBM Tivoli Access Manager for e-business Command Reference*.

**Web Portal Manager**
Web Portal Manager is an optional Web-based graphical user interface (GUI) used for Tivoli Access Manager administration. Web Portal Manager provides a GUI interface that allows you to manage users, groups, roles, permissions, policies, domains, and servers, as well as to perform other tasks.

Web Portal Manager is available only on the AIX, Solaris, and the Microsoft Windows operating systems. This optional GUI must be installed separately using the Tivoli Access Manager Web Portal Manager CD for your platform. A key advantage to using Web Portal Manager is that you can perform these tasks remotely using any supported Web browser — without any special network configuration.

For specific task information, refer to the task specific chapters in this guide. For more information on using Web Portal Manager, see the Web Portal Manager online help.

**Administration API**
The administration API component of Tivoli Access Manager provides a set of programming interfaces that allow you to write applications to manage users, groups, rules, permissions, policies, domains, and servers. Both C and Java language versions of these functions are available.

Details on the administration API are provided in the *IBM Tivoli Access Manager for e-business Administration C API Developer Reference* and the *IBM Tivoli Access Manager for e-business Administration Java Classes Developer Reference*.

### Security policy overview

The goal of any security policy is to adequately protect business assets and resources with a minimal amount of administrative effort. First, you must define what resources need to be protected. These could be any type of data object, such as files, directories, network servers, messages, databases, or Web pages. Then, you must decide what users and groups of users should have access to these protected resources. You also need to decide what type of access to these resources should be permitted. Finally, you must apply the proper security policy on these resources to ensure that only the right users can access them.

The enforcement of the security policy is the job of the resource manager. The resource manager calls the Tivoli Access Manager authorization service with the
credentials of the user making the request, the type of access desired, and the object to be accessed. The credential provides detailed information, acquired during authentication, that describes the user, any group associations, and other security-related identity attributes. Credentials can be used to perform a multitude of services, such as authorization, auditing, and delegation.

The authorization service, also known as the authorization engine, uses the security policy to determine whether the request should be allowed, denied, or conditionally allowed pending additional verification by the resource manager. The resource manager takes the recommendation of the authorization service, performs any additional verification actions, and ultimately either denies the request, or permits the request to be processed.

For example, suppose that Todd wants to access a particular Web page that is located on a Web site protected by IBM Tivoli Access Manager WebSEAL. WebSEAL is a resource manager that is responsible for managing and protecting Web-based information and resources and must decide whether or not Todd can access that page. The resource manager obtains the credentials for Todd, and then asks the authorization service whether Todd has read access to the Web page. The authorization service checks the security policy and determines that Todd should be permitted access, so it recommends to the resource manager that the request be granted. The resource manager then directs Todd’s request to the appropriate back-end Web server, which provides the Web page.

The security policy in Tivoli Access Manager is defined through the use of access control lists (ACLs), protected object policies (POPs), and authorization rules.

**Authorization API standard**

Authorization services are a critical part of an application’s security architecture. After a user passes the authentication process, authorization services proceed to enforce the business policy by determining what services and information the user can access.

For example, a user accessing a Web-based retirement fund would be able to view personal account information after an authorization server verifies the identity, credentials, and privilege attributes of that user.

The standards-based authorization API (aznAPI) allows applications to make calls to the centralized authorization service; thereby, eliminating the necessity for developers to write authorization code for each new application.

The authorization API allows businesses to standardize all applications on a trusted authorization framework. With the authorization API, businesses can provide more control over access to resources on their networks.

**Authorization: conceptual model**

When servers enforce security in a domain, each client must provide proof of its identity. In turn, security policy determines whether that client is permitted to perform an operation on a requested resource. Because access to every resource in a domain is controlled by a server, the server’s demands for authentication and authorization can provide comprehensive network security.

In security systems, authorization is distinct from authentication. Authorization determines whether an authenticated client has the right to perform an operation
on a specific resource in a domain. Authentication ensures that the individual is who he claims to be, but says nothing about the rights to perform operations on a protected resource.

In the Tivoli Access Manager authorization model, authorization policy is implemented independently of the mechanism used for user authentication. Users can authenticate their identity using either public/private key, secret key, or customer-defined mechanisms.

Part of the authentication process involves the creation of a credential that describes the identity of the client. Authorization decisions made by an authorization service are based on user credentials.

The resources in a domain receive a level of protection as dictated by the security policy for the domain. The security policy defines the legitimate participants of the domain. It also defines the degree of protection that is surrounding each resource requiring protection.

The basic components of the authorization process, as shown in Figure 1, include:

- A resource manager responsible for implementing the requested operation when authorization is granted.
  A component of the resource manager is a policy enforcer that directs the request to the authorization service for processing.
- An authorization service that performs the decision-making action on the request.

![Figure 1. General authorization model](image)

Traditional applications bundle the policy enforcer and resource manager into one process. Examples of this structure include Tivoli Access Manager WebSEAL, Tivoli Access Manager for Operating Systems, and Tivoli Access Manager for Business Integration.

The independent functionality of these authorization components allows flexibility in the design of the security enforcement strategy.

For example, such independence allows the security administrator to control:

- Where the processes are located
The benefits of a standard authorization service

Authorization in most systems, both legacy and new, is tightly coupled to individual applications. Companies typically build applications over time to serve their business needs. Many of these applications require some specific form of authorization.

The result is often a wide variety of applications with differing authorization implementations. These proprietary authorization implementations require separate administration, are difficult to integrate, and result in higher costs of ownership.

A distributed authorization service can provide these independent applications with a standard authorization decision-making mechanism. Benefits of such a standard authorization service include:

- Reduced cost of developing and managing access to applications
- Reduced total cost of ownership and management of separate authorization systems
- Leverage of existing security infrastructure
- Allow new businesses to open more securely
- Enable newer and different kinds of applications
- Allow shorter development cycles
- Share information securely

Tivoli Access Manager authorization service overview

Tivoli Access Manager can be integrated into existing legacy and emerging infrastructures to provide a secure, centralized policy management capability. The Tivoli Access Manager authorization service, together with resource managers, provides a standard authorization mechanism for business network systems, as shown in Figure 2.

![Figure 2. Tivoli Access Manager server components](image-url)
Existing applications can take advantage of the authorization service. Authorization policy is based on user or group roles and can be applied to network servers, individual transactions or database requests, specific Web-based information, management activities, and user-defined objects.

The authorization API allows existing applications to make calls to the authorization service which in turn makes decisions based on the corporate security policy. For more information on authorization API, see “The Tivoli Access Manager authorization API” on page 13.

The Tivoli Access Manager authorization service is also extensible and can be configured to call on other authorization services for additional processing using the external authorization service plug-in interface.

The authorization service provides the following benefits:

- The service is application independent.
- The service uses a standard authorization coding style that is language independent (the authorization API).
- The service is centrally managed and therefore easy to administer — the addition of a new employee, for example, requires modifying the privilege database in one central location, rather than across multiple systems.
- The service addresses the application of security services in a heterogeneous cross-platform environment.
- The service integrates existing non-Tivoli Access Manager authorization systems through an external authorization service capability.
- The service has a scalable and flexible architecture that can be easily integrated with existing infrastructure.
- The service enables multi-tiered authorization — a credentials packet can be passed through the multiple layers of an application process or transaction.
- The service uses a common and effective auditing model.
- The service is independent of any authentication mechanism.

The Tivoli Access Manager authorization service

The Tivoli Access Manager authorization service is responsible for the authorization decision-making process that helps to enforce a network security policy. Authorization decisions made by the authorization service result in the approval or denial of client requests to perform operations on protected resources in a domain.

Components

The authorization service is made up of three basic components:

- Master authorization policy database
- Policy server
- The authorization decision-making evaluator

Policy database

The policy database, also referred to as the master authorization policy database and the master authorization database, contains the security policy information for all resources in a domain. Each domain has its own policy database. The contents of this database are manipulated using the Web Portal Manager, the pdadmin command line utility, and the administration API.
**Policy server**
The policy server (pdmgrd) maintains the policy databases, replicates this policy information throughout the domains, and updates the database replicas whenever a change is made to the master.

The policy server also maintains location information about the other Tivoli Access Manager and non-Tivoli Access Manager resource managers operating in the domain.

**Authorization evaluator**
The authorization evaluator is the decision-making process that determines a client’s ability to access a protected resource based on the security policy. The evaluator makes its recommendation to the resource manager which, in turn, responds accordingly.

Registry database replication parameters are configurable for each evaluator.

Figure 3 illustrates the main components of the authorization service:

![Figure 3. Authorization service components](image)

**Authorization service interfaces**
The authorization service has two interfaces where interaction takes place:

- **Management interface** — The security administrator manages the security policy by using the Web Portal Manager or the pdadmin command line utility to apply policy rules on resources in a domain. The security policy is managed in the policy database by the policy servers.
  This interface is complex and involves detailed knowledge of the object space, policies, and credentials.

- **Authorization API** — The authorization API passes requests for authorization decisions from the resource manager to the authorization evaluator which then passes back a recommendation whether the request should be granted or denied.

**Replication for scalability and performance**
Authorization service components can be replicated to increase availability in a heavy-demand environment.
You can configure the master authorization policy database, containing policy rules and credential information, to automatically replicate. Resource managers that call the authorization service have two options for referencing this database information:

- The application, when configured to work seamlessly with the authorization evaluator, uses a local cache of the database. The database is replicated for each resource manager that uses the authorization service in local cache mode.
- The application uses a shared replica cached by the remote authorization server component. The database is replicated for each instance of the authorization server. Many applications can access a single authorization server.

Update notification from the policy server (whenever a change has been made to the master authorization policy database) triggers the caching process to update all replicas, as shown in Figure 4:

Figure 4. Replicated authorization service components

**Performance notes**

- In addition to update notifications direct from the policy server, the resource managers can be configured to also check the version of the master authorization policy database every few minutes to ensure they have not missed an update notification. This is known as polling and is not enabled by default. If an update notification fails to reach a server, a log entry is created. In both cases a retry mechanism also ensures the update happens in the future.
- The cached authorization policy information results in high system performance. For example, when WebSEAL does an authorization check, it checks the policy in its own cached version of the database. WebSEAL does not have to access the network to obtain this information from the master database. The result is very fast response times (performance) for authorization checks.
- Individual authorization results are not cached by the calling application server.
Implementing a network security policy

The security policy for a domain is determined by controlling user and group participation in the domain and applying rules to resources requiring protection. These rules are defined through the use of Access control lists (ACLs), protected object policies (POPs), and authorization rules. The authorization service enforces these policies by matching a user’s credentials with the permissions in the policy assigned to the requested resource. The resulting recommendation is passed to the resource manager, which completes the response to the original request.

Defining and applying security policy

You protect system resources by defining a security policy. This security policy is created by defining access control lists (ACLs), protected object policies (POPs), and authorization rules, and then applying these policies to the object representations of those resources in the object space. You can apply ACLs, POPs, and authorization rules to the same object. The pdadmin command line utility, the Web Portal Manager GUI, and the administration API are used to define this policy.

The authorization service performs authorization decisions based on the policies applied to these objects. When a requested operation on a protected object (also referred to as a protected resource) is permitted, the resource manager responsible for the resource implements this operation.

One policy can dictate the protection parameters of many objects. Any change to the security policy affects all objects to which the policy is attached.

Explicit and inherited policy

A security policy can be explicitly applied or inherited. The Tivoli Access Manager protected object space supports inheritance of ACLs, POPs, and authorization rules. This is an important consideration for the security administrator who manages the object space. The administrator needs to apply explicit policies only at points in the hierarchy where the rules must change, as shown in Figure 5.

![Figure 5. Explicit and inherited policies](image)

Examples of policy types include:

- Hard-coded rules
- External authorization capability
- Special secure labeling
• Access control lists (ACLs), protected object policies (POPs), and authorization rules

**Access control lists**
An access control list (ACL) policy, or ACL policy, is the set of actions, controls, or permissions that specifies the conditions necessary for a particular user or group to perform certain operations on that resource. ACL policy definitions are important components of the security policy established for a domain.

An ACL policy specifically determines what operations can be performed on a resource, and who can perform those operations. An ACL policy is made up of one or more entries that include user and group designations and either their specific permissions or rights.

**Protected object policies**
Protected object policies (POPs) contain additional conditions that must be met in order to be granted access. Unlike ACLs, which are dependent on what user or group is attempting the action, POPs affect all users and groups. POPs also indicate whether requests should be audited. It is the responsibility of Tivoli Access Manager and the resource manager to enforce the POP conditions.

**Authorization rules**
Authorization rules are defined to specify further conditions that must be met before access to a resource is permitted. Rules allow you to make authorization decisions based on the context and the environment surrounding the request, as well as who is attempting the access, and what type of action is being attempted. These conditions are evaluated as a Boolean expression to determine if the request should be allowed or denied.

**The authorization process: step-by-step**

Figure 6 illustrates the complete authorization process:

![Figure 6. The Tivoli Access Manager authorization process](image-url)
Notes:
1. An authenticated client request for a resource is directed to the resource manager server and intercepted by the policy enforcer process. For example, the resource manager can be WebSEAL for Hypertext Transfer Protocol (HTTP), HTTPS access or another application.
2. The policy enforcer process uses the authorization API to call the authorization service for an authorization decision. For more information on the authorization API, see "The Tivoli Access Manager authorization API."
3. The authorization service performs an authorization check on the resource. See page 37 for details on the algorithm used.
4. The decision to accept or deny the request is returned as a recommendation to the resource manager (through the policy enforcer).
5. If the request is finally approved, the resource manager passes the request on to the application responsible for the resource.
6. The client receives the results of the requested operation.

The Tivoli Access Manager authorization API

The Tivoli Access Manager authorization application programming interface (API) allows Tivoli Access Manager applications and other applications to query the authorization service to make authorization decisions.

The authorization API is the interface between the resource manager (requesting the authorization check) and the authorization service itself. The authorization API allows Tivoli Access Manager resource managers and other resource managers to ask for an authorization decision, but shields the application from the complexities of the actual decision-making process.

The authorization API provides a standard programming model for coding authorization requests and decisions. The authorization API lets you make standardized calls to the centrally managed authorization service from any legacy or newly developed application.

The authorization API can be used in one of two modes:

- **Remote cache mode**
  In this mode, the API is initialized to call the (remote) authorization server (pdacld) to perform authorization decisions on behalf of the application. The authorization server maintains its own cache of the replica authorization policy database. This mode is best suited for handling authorization requests from application clients.
  For more information on remote cache mode, see “Authorization API: remote cache mode” on page 15.

- **Local cache mode**
  In this mode, the API is initialized to download and maintain a local replica of the authorization database for the application. Local cache mode provides better performance because the application performs all authorization decisions locally instead of across a network. However, the overhead of database replication and the security implications of using this mode make it best suited for use by trusted application servers.
  For more information on local cache mode, see “Authorization API: local cache mode” on page 16.
One of the primary values and benefits of the authorization API is its ability to shield the resource manager from the complexities of the authorization service mechanism itself. Issues of management, storage, caching, replication, credential formats, and authentication methods are all hidden behind the authorization API.

The authorization API also works independently from the underlying security infrastructure, the credential format, and the evaluating mechanism. The authorization API makes it possible to request an authorization check and get a simple yes or no recommendation in return. The details of the authorization check mechanism are invisible to the user.

**Using the authorization API: two examples**

Applications can use the authorization API to perform access control on very specific and specialized processes.

**Example 1:**
A graphical user interface can be designed to dynamically show task buttons as active or inactive, according to the results of the authorization check.

**Example 2:**
Another use of the authorization API is demonstrated in [Figure 7 on page 15](#) illustrating a request for a Common Gateway Interface (CGI) transaction by a Web application.

The lowest level of authorization, as illustrated in Figure A of [Figure 7 on page 15](#) involves an “all-or-nothing” access control on the uniform resource locator (URL). This coarse-grained level of authorization only determines if the client can run the CGI program. If access is allowed to the CGI application, no further control is available to resources manipulated by the CGI application.

As illustrated in Figure B of [Figure 7 on page 15](#) access controls have been set on resources that the CGI program manipulates. The Web application is configured to use the authorization API. Now the CGI program can call the authorization service to make authorization decisions on the resources it manipulates — based on the identity of the requesting client.
Authorization API: remote cache mode

In remote cache mode, resource managers use the function calls provided by the authorization API to communicate to the (remote) authorization server (pdacld). The authorization server functions as the authorization decision-making evaluator and maintains its own replica authorization policy database.

The authorization server makes the decision and returns a recommendation to the application through the API. The server can also write an audit record containing the details of the authorization decision request.

There must be an authorization server running somewhere in a domain when using remote cache mode, as shown in Figure 8 on page 16. The authorization server can be located on the same machine as the application or on another machine. You can also install the authorization server on more than one machine in a domain to allow for high availability. The authorization API transparently performs failover when a particular authorization server fails.
Authorization API: local cache mode

In local cache mode, the API downloads and maintains a replica of the authorization policy database on the resource manager’s local file system. It performs all authorization decisions in-memory, which results in higher performance and better reliability.

The local replica is persistent across invocations of the application. When the API starts in replica mode, it checks for any updates to the master authorization policy database that might have occurred since the local replica was built.
External authorization capability

In some situations, the standard Tivoli Access Manager policy implementations—ACLs, POPs, and authorization rules might not be able to express all the conditions required by an organization’s security policy. Tivoli Access Manager provides an optional external authorization capability to accommodate any additional authorization requirements.

The external authorization service allows you to impose additional authorization controls and conditions that are dictated by a separate, external, authorization service module.

Extending the authorization service

External authorization capability is automatically built into the Tivoli Access Manager authorization service. If you configure an external authorization service, the Tivoli Access Manager authorization service simply incorporates the access decision paths into its evaluation process.

Resource managers that use the authorization service, such as WebSEAL and any application using the authorization API, benefit from the additional, but seamless, contribution of a configured external authorization service. Any addition to the security policy through the use of an external authorization service is transparent to these applications and requires no change to the applications.

The external authorization service architecture allows the full integration of an organization’s existing security service. An external authorization service preserves
a company’s initial investment in security mechanisms by allowing legacy servers to be incorporated into the Tivoli Access Manager authorization decision-making process.

**Imposing conditions on resource requests**

An external authorization service can be used to impose more specific conditions or system-specific side effects on a successful or unsuccessful access attempt.

Examples of such conditions include:

- Causing an external auditing mechanism to record the successful or unsuccessful access attempt
- Actively monitoring the access attempt and causing an alert or alarm whenever unacceptable behavior is detected
- Conducting billing or micro-payment transactions
- Imposing access quotas on a protected resource

**The authorization evaluation process**

An authorization decision that incorporates an external authorization server takes place in the following manner:

1. If a trigger condition is met during the course of an access decision, the external authorization services that have been configured for that condition are each called in turn to evaluate their own external authorization constraints.

   Invocation of the external authorization service occurs regardless of whether or not the necessary permission is granted to the user by the Tivoli Access Manager authorization service.

2. Each external authorization service returns a decision of permitted, denied, or indifferent.

   When indifferent is returned, the external authorization service has determined that its functionality is not required for the decision process and that it does not participate.

3. Each external authorization service decision is weighted according to the level of importance that its decision carries in the process.

   The weighting of individual external authorization services is configured when the service plug-in is loaded.

4. All authorization decision results are summed and combined with the decision made by the Tivoli Access Manager authorization service. The resulting decision is returned to the caller.

**Example**

[Figure 10 on page 19](#) illustrates an authorization decision involving an application server and an external authorization service.
In this example, the purpose of the external authorization service is to impose a quota restriction on how often a photo-quality printer resource can be accessed.

The service implementation imposes a limit on the number of job submissions that any one person can make to this printer in one week. An external authorization service trigger condition has been attached to the photo printer resource so that the external authorization service is invoked anytime that the photo printer is accessed.

The external authorization service has been loaded with the default decision weighting of 101, which overrides any decision made by the Tivoli Access Manager authorization service, should it need to do so.

1. The resource manager server receives a request from a client for access to an online photo printing resource. The client is a member of the appropriate group GraphicArtists and so is normally permitted to submit jobs to the printer.

2. The application server first consults the Tivoli Access Manager authorization service to determine whether the requesting user has permission to submit jobs to the printer.

3. The authorization service checks the access permissions on the target requested object and compares these with the capabilities of the requesting user:

   group GraphicArtists rx

   In the ACL on the printer resource, the x permission grants any user in the GraphicArtists group access to the resource. Therefore, the authorization service grants the user permission to submit the job.

4. Because the photo printer resource is being accessed and an external authorization service trigger condition was attached to this object, a request is also made to the external authorization service configured for that trigger condition.

*Figure 10. External authorization service with an application server*
The external authorization service receives all of the Access Decision Information (ADI) that was passed in with the original access decision check by the resource manager server.

5. The external authorization service consults a record of previous accesses made by this user. If the requesting user has not exceeded the quota for the week, it returns an access decision of “indifferent.”

The implication is that the external authorization service is indifferent to the request and has no intention of participating in the access decision because its conditions for denying access have not been met.

However, if the user has exceeded the quota, then the external authorization service returns a decision of “access denied.”

For this example, it is assumed that the requester has exceeded the quota and that the external authorization service detects this and returns an “access denied” decision.

6. The Tivoli Access Manager authorization service receives the “access denied” result from the external authorization service. It then takes this decision and weights it with the default external authorization service weighting value of 101.

The results of the external authorization service decision and the decision made by the Tivoli Access Manager authorization service are combined. The result is “access denied” because the result of the external authorization service (-101) outweighs that of the Tivoli Access Manager authorization service (100).

7. The resource manager server rejects the job submission to the photo printer resource.

8. The resource manager server returns a response to the caller to indicate that the job was rejected.

**Implementing an external authorization service**

Two general steps are required to set up an external authorization service:

1. Write an external resource manager service plug-in module with an authorization interface that can be referenced during authorization decisions.

2. Register the external authorization service with the resource manager so that the resource manager can load the plug-in service at initialization time.

Registering the service sets a trigger condition for the invocation of the external authorization service. When the trigger condition is encountered during an authorization check, the external authorization service interface is invoked to make an additional authorization decision.

**Deployment strategies**

Tivoli Access Manager allows you to implement an external authorization service in several ways:

- Any number of external authorization services can be registered with resource manager applications. Applications that can load external authorization services include the authorization server (pdacld), other Tivoli Access Manager resource managers, and any other resource manager applications that you create.
- Remote-mode authorization API clients, which make requests to the authorization server for authorization decisions, automatically make use of any external authorization service that is loaded by the authorization server.
- More than one external authorization service can be called for any single trigger condition. In this case, the results of each external authorization service is
weighted accordingly, and then the results are combined with the result of the Tivoli Access Manager authorization service.

- Trigger conditions can be placed on objects, using a POP trigger, such that any request to an object, regardless of the operation that is being requested, triggers a call to the external authorization services that are configured for the trigger.
- Trigger conditions can also be placed on the operations requested by a user. For example, an external authorization service can be triggered specifically when a user requests a Write operation to a protected resource, but not for any other operation. It is then possible to develop sets of operations for which one or more external authorization services in combination are triggered according the set of operations requested.
- The external authorization services are implemented as dynamically loadable library (dynamic link library (DLL)) modules. This greatly simplifies the task of external authorization service development. There is no requirement to make remote requests to the external authorization service and the overhead of making the call is equivalent to the overhead of a function call.
- The combination of the authorization API and an external authorization service provides a highly extensible and flexible solution for implementing a complex security policy.
Chapter 2. Web Portal Manager

Tivoli Access Manager provides two types of user interfaces:

- The Web Portal Manager, a Web-based interface
  This interface is used to provide management and administration of domains, users, groups, roles, permissions, access control lists, protected object policies, authorization rules, protected object spaces, protected objects, and other resources in your secure domain.

- The **pdadmin** command, a command line interface
  This interface also provides the similar tasks for managing security policy for the secure domain.

The command line utility is installed as part of the Tivoli Access Manager runtime package. You can also automate certain management functions by writing scripts that use **pdadmin** commands. The *IBM Tivoli Access Manager for e-business Command Reference* provides detailed information about the **pdadmin** command line interface and other command line utilities.

One of the factors to keep in mind when deciding which interface to use is that by using the Web Portal Manager, you can perform only a large subset of the **pdadmin** commands. To compare the mapping that exists between the administration **pdadmin** command line interface (CLI) and the Web Portal Manager interface, see Appendix C, “Administration command line and Web Portal Manager equivalents,” on page 329.

Another difference between the two types of interface is that, by using the **pdadmin** command, you can specify a file. Using the Web Portal Manager, you cannot specify a file name, but in some case you can cut and paste to use the contents of the file (for example, authorization rules).

This chapter contains the following sections:

- “Types of administration”
- “Delegate administration functions” on page 24
- “Self-care” on page 24
- “Self-registration” on page 24

Web Portal Manager procedures are included for:

- “Web Portal Manager common tasks” on page 25
- “Web Portal Manager interface customization” on page 28
- “Self-registration tasks” on page 29

### Types of administration

Tivoli Access Manager provides two types of administration:

- Web Portal Manager administration
- Web Portal Manager delegate administration

Web Portal Manager administration provides a Web-based interface similar to the base **pdadmin** functions. Web Portal Manager administration does not include the delegate administration functions.
Delegate administration functions

Web Portal Manager delegate administration provides a Web-based interface that includes a set of delegated management services. The Web Portal Manager delegate administration functions are considered to be a separate application from the administration functions. The delegated management services enable a business to delegate user administration, group and role administration, security administration, and application access provisioning to participants (subdomains) in the business system. These subdomains can further delegate management and administration to trusted subdomains under their control; thereby supporting multi-level delegation and management hierarchy based on roles.

The delegate administration supports:
- Creation of multiple enterprise domains
- Assignment of users to be domain administrators
- Assignment of administrator types (such as: Tivoli Access Manager Administrator, Domain Administrator, Senior Administrator, Administrator, and Support Administrator) and enforcement of the administrative functions that can be performed with each administrator type
- Use of self-registration, meaning to become a registered Tivoli Access Manager user without the involvement of an administrator, and self-care to reduce the administration load.

Refer to Chapter 16, “Delegated administration,” on page 155 for a complete description of Tivoli Access Manager delegate administration functions.

Self-care

Web Portal Manager deployments can grow to support large number of users. As the number of users grows, so does the number of administrators required to manage these users. Self-registration and self-care are features of the Web Portal Manager that can be used to reduce the administration load.

The Web Portal Manager supports self-care operations by allowing Tivoli Access Manager users to change their Tivoli Access Manager password through the Web Portal Manager. Users can go to the Web Portal Manager delegate administration page and manage their passwords. After logging in, the user should go to the Change My Password task.

Self-registration

Self-registration is the process by which a user can enter required data to become a registered Tivoli Access Manager user, without the involvement of an administrator.

The Web Portal Manager includes a sample application that allows end-users to perform self-registration. Note that this sample is supported only on an LDAP registry, not Domino or Active Directory.

Included with the Web Portal Manager is a sample code that implements a self-registration page. The sample code shows how to use the Tivoli Access Manager Java Administration APIs along with Java 2 Platform, Enterprise Edition (J2EE) servlets and Java Server Pages (JSPs) to implement self-registration. See “Self-registration tasks” on page 29.
Web Portal Manager common tasks

This chapter provides procedures for the more common Web Portal Manager tasks, such as:

- “Start Web Portal Manager administration”
- “Start Web Portal Manager delegate administration”
- “Log in and sign off” on page 26
- “Get online help” on page 26
- “Search” on page 26
- “View lists” on page 27

Start Web Portal Manager administration

Prior to starting Web Portal Manager, ensure the IBM WebSphere Application Server is running. Then, use one of the following URLs to start Web Portal Manager administration:

- If you have installed, configured, and enabled SSL, type the following URL in your Web browser:
  
  https://hostname/pdadmin

  where hostname is the machine where IBM HTTP server and WebSphere are running.

- If you do not have SSL installed, configured, and enabled, type the following URL in your Web browser:
  
  http://hostname/pdadmin

For example:

https://testgroup.austin.ibm.com/pdadmin

Start Web Portal Manager delegate administration

The Web Portal Manager delegate administration functions are accessed from a different URL than the Web Portal Manager base pdadmin command functions. The Web Portal Manager delegate administration functions are considered to be a separate application.

Use one of the following URLs to start Web Portal Manager delegate administration and access the Web Portal Manager base pdadmin functions:

- If you do not have SSL installed, configured, and enabled, type the following URL in your Web browser:
  
  http://hostname/delegate

- If you have installed, configured, and enabled SSL, type the following URL in your Web browser:
  
  https://hostname/delegate

  where hostname is the machine where IBM HTTP server and WebSphere are running.

For example:

https://testgroup.austin.ibm.com/delegate
Users are able to go to the Web Portal Manager delegate administration page and manage their passwords. After logging in, the user should go to the **Change My Password** task.

**Log in and sign off**

To log in to Web Portal Manager:

1. Start Web Portal Manager administration.
2. Provide Web Portal Manager authentication, such as a user name and password.
3. After the Tivoli Access Manager logo appears, select and perform tasks, as needed.
4. Click **Sign Off** on the bottom status bar to shut down Web Portal Manager administration.

**Get online help**

Instructions for completing tasks using the Web Portal Manager are documented in the online help system. Reference the help system when entering information in fields, or when selecting or deselecting choices for Web Portal Manager.

To use online help for Web Portal Manager:

1. Log in to the domain.
2. Select a task such as **Group → Import Group**.
3. Click on the question mark icon on the right side of the window, in the task title bar.
   A help window displays that provides online information for completing the task.
4. Close the help window when the task has been completed.

**Search**

Use the search function to display a list of items (such as user IDs or group names) matching the search criteria that you specify.

To search using Web Portal Manager:

1. Log in to the domain.
2. Select a search task. For example, click **User → Search Users**.
3. Type a user ID or use the wildcard character (*) to limit the search. The wildcard character (*) can appear anywhere in the user ID.
   For example:
   - To search for all user IDs, type *. This search criteria would find ibm, IBM, and Ibm because the search is case insensitive.
   - To search for user IDs beginning with the letters ibm, type ibm*.
   - To search for user IDs ending with the letters ibm, type *ibm.
   - To search for user IDs beginning with the letters ib and ending with the letters m, type ib*m. This search criteria would find ibm, IBM, and Ibm because the search is case insensitive. It would also find IB.com and Ibstam.
4. Type the maximum number of user IDs or group names that you want to be displayed.
   Note that typing a number higher than the default value of 100 can result in slow response times.
5. Close the help window when the task has been completed.

View lists
Use the list function to display a table of items matching the search criteria that you specify.

To view lists of items using Web Portal Manager:
1. Log in to the domain.
2. Select a list task. For example, click **ACL → List ACL** to view a table listing all ACLs.
   From the list, you can create a new ACL, delete one or more existing ACLs, view properties of a selected ACL, change options, and change filters.

Change options
To change viewing options using Web Portal Manager:
1. Log in to the domain.
2. Select a list task. For example, click **ACL → List ACL** to view a table listing all ACLs.
3. Click **Options** to toggle and show the **Entries Per Page** field.
   For user IDs, the **Show Entry Details** check box is also displayed. When this check box is selected, you see these additional columns in the table:
   - First Name
   - Last Name
   - Password Valid
   - Account Valid
   Unselect the check box to remove these columns from the view.
4. To display a specified number of entries per page, accept the default value or type the number in the **Entries Per Page** field, and then click **OK**.
5. When the number of entries per page is exceeded, more than one page of entries is displayed, additional navigation items are added to the end of the table on each page. You can navigate between pages by clicking the page toggle icon, or by typing a page number and then clicking **Go**. The total number of entries in the table is also displayed (for example, **Total: 65, Filtered: 65**).
6. Click **Close** to hide the **Entries Per Page** field.

Change filters
To change filtering using Web Portal Manager:
1. Log in to the domain.
2. Select a list task. For example, click **ACL → List ACL** to view a table listing all ACLs.
3. Click **Filters** to toggle and show the filter link. If no filters are used, a **None** link is displayed for the column.
4. Click the filter link.
   For user IDs, each column displays its own filter link.
5. In the **Text** field, type the text that you want to use for filtering, such as Luca.
6. Select one of the filtering criteria from the list:
   - Contains
   - Starts with
• Ends with

7. Click OK to start the filtering process.
   When the number of entries per page is exceeded, more than one page of entries is displayed, additional navigation items are added to the end of the table on each page. You can navigate between pages by clicking the page toggle icon, or by typing a page number and then clicking Go. The total number of entries and the number of filtered entries in the table are also displayed (for example, Total: 65, Filtered: 65).

8. Click Close to hide the filtering options. The filtering link changes from None to the text you entered for filtering (for example, Luca).

**Web Portal Manager interface customization**

The Web Portal Manager allows for customization of the interface for a Tivoli Access Manager user. A Tivoli Access Manager user can rebrand the Web Portal Manager by modifying the configuration to specify which HTML page or GIF file should be loaded when the Web Portal Manager starts up.

**Customize the images**

To customize the images:

1. Change the value of the image options in the pdwpm.conf configuration file to specify different images.
   There are four images areas that can be customized in the Web Portal Manager by changing the options:
   a. loginGif — This shows the image on the login page.
   b. splashGif — This shows the image on the welcome page, after the login page.
   c. infoBarGif — This shows the IBM image on the bottom right of the page.
   d. bannerFile — This shows the banner at the top of each page.

2. Place the new images in one of the following directories:
   • For UNIX systems:
     `websphere_install_dir/WebSphere/AppServer/installedApps/
     server_name/TAMWPM.ear/padmin.war/images`
   • For Windows systems:
     `Program Files\WebSphere\AppServer\installedApps\server_name\TAMWPM.ear\padmin.war\images`

3. For locale-specific versions of the images, create subdirectories under the following directories for each of the locales and place the new images in the subdirectories:
   • For UNIX systems:
     `websphere_install_dir/WebSphere/AppServer/installedApps/
     server_name/TAMWPM.ear/padmin.war/images/locale/myImage.gif`
   • For Windows systems:
     `Program Files\WebSphere\AppServer\installedApps\server_name\TAMWPM.ear\padmin.war\images\locale\myImage.gif`

**Create a customized top banner**

To create a customized top banner for the Web Portal Manager:

1. Create a JSP or HTML file.
2. Put the JSP or HTML file you created in the same directory as the `top_banner.jsp` file.
3. Change the value of the **bannerFile** option, whose default value is `bannerFile=top_banner.jsp`, to point to the new JSP or HTML filename.

The layout of the frame is set up by the `pdmainframe.jsp` file, which sets the frame for the page to be a height of 50 pixels.

### Self-registration tasks

Tivoli Access Manager provides a self-registration sample to demonstrate how it works.

**Note:** This sample is supported only for an LDAP user registry, not IBM Lotus Domino or Microsoft Active Directory user registry.

### Perform self-registration

One possible scenario for implementing self-registration is where a user opens a Web browser to view a self-registration Web page. On this Web page, the user enters specific identification information (either company-specific or user-specific) with a Tivoli Access Manager user ID and password. The identification information provided by the user is then validated and the user is created in the Tivoli Access Manager registry.

Because users do not usually have permission to create objects in Tivoli Access Manager, the self-registration sample requires the ID and password of an administrator who has permission to create users. This login information is then used to create users when somebody enters the required information on the registration page.

The following information is requested the first time the self-registration sample is accessed. This data is saved by the servlet in memory and then used to create users who request to be registered.

- Administrator name
- Password
- Registry container

The administrator name and password should be the name of an administrator who has permission to create users in Tivoli Access Manager. The sec_master administrator has the proper access by default. The Registry Container field should be the base name in LDAP where user entries should go. This value is used to construct the distinguished name (DN) of self-registered users.

For example, enter `o=ibm,c=us` and the registered users are created in LDAP as, `cn=FirstnameLastname,o=ibm,c=us`. The user is not added to any groups. In a real application, the user would probably be added to some groups to have access to some applications. After the administrator information is entered, this page is not shown again. If you access the sample, you are shown only the registration page where you can enter the first name, last name, and a password.

Note that the administrator login is saved in the servlet session. Any user who accesses the self-registration sample from the same browser can create a user in Tivoli Access Manager. You must restart the application server to clear the administrator login information.
For this sample, the ID and password are not saved in a secure manner. If you use this sample as the basis for a production registration application, you should consider ways to secure the administrator login information.

**Change Java server pages**

There are three JSP pages in the sample application:

- `regAdmin.jsp` is the page shown to gather the administrator login information.
- `regProp.jsp` is the page shown to gather user first name, last name, and password.
- `regControl.jsp` contains the code that creates the user. This page receives and processes the registration requests. This could also be a servlet class.

The files are installed in the following directory:

- For UNIX systems:
  
  ```
  websphere_install_dir:/WebSphere/AppServer/installedApps/
  server_name/TAMWPM.ear/register.war/register
  ```

- For Windows systems:
  
  ```
  websphere_install_dir:\Program Files\WebSphere\AppServer\installedApps\
  server_name\TAMWPM.ear\register.war\register
  ```

where `websphere_install_dir` is the directory where WebSphere is installed.

When the administrator login information is entered, a JRTE `PDContext` is created and stored in the user servlet session as shown in the following:

```java
String adminid = request.getParameter("admin");
String adminPassword = request.getParameter("password");
String ldapSuffix = request.getParameter("suffix");
...
// Try a login
try {
    ctx = new PDContext(adminid,
            adminPassword.toCharArray(),
            url);
    // Save the PDcontext and the LDAP Suffix
    session.setAttribute("regAdminCtx", ctx);
    session.setAttribute("ldapSuffix", ldapSuffix);
} catch(PDException e) {
    // process exception ...
}
```

After the user enters the new user information, the `PDContext` is retrieved from the session and used to create the new user as shown in the following:

```java
// Creating the PD User
pwd = request.getParameter("password");
ldapcn = request.getParameter("ldapcn");
ldapsn = request.getParameter("ldapsn");
ldapdn = "cn=" + ldapcn + ldapsn + "," + ldapSuffix;
userid = ldapcn + ldapsn;
desc = ldapcn + " " + ldapsn;
ctx = (PDContext)session.getAttribute("regAdminCtx");
// Make sure the session has not timed out
if ( ctx == null ) {
    %>
    <%@ include file="regAdmin.jsp" %>
    <%
    return;
    }
```
PDMessages messages = new PDMessages();
try {
    createUser(bundle, ctx, userid, pwd, desc, ldapcn,
    ldapsn, ldapdn, usergroups, acc_valid,
    pwd_valid, gso_user, no_pwd_pol,
    messages);
    succmsg = userid +
    ResourceFile.getString(bundle,
    "userRegisteredMsg");
} catch(PDException e) {
    // process exception
    ...
}

The new user's ID is the first name and last name concatenated together.
Chapter 3. Tivoli Access Manager administration

The administration of Tivoli Access Manager involves the following major tasks:
1. Create a domain, if necessary.
2. Install and configure resource managers. All of the Tivoli Access Manager resource managers, such as WebSEAL and other blade components such as Plug-in for Web Server, automatically create a protected object space and create the required protected resources (also known as protected objects) when they are configured.
   • If necessary, create an object space.
   • If necessary, define protected objects in the object space that represent the resources that are to be protected.
3. Define users and groups that require access to the protected resources.
4. For the protected objects, you can define:
   • Who is allowed access.
   • What type of access is permitted.
   • When that access is allowed.
   • What other conditions must be met to permit access.
   • Whether the access request is to be audited.
5. Implement your security policy by attaching an access control list (ACL), a protected object policy (POP), and an authorization rule to objects in the protected object space.

Domains

A domain consists of all the resources that require protection along with the associated security policy used to protect those resources. These resources can be any physical or logical entity, including objects such as files, directories, Web pages, printer and network services, and message queues. Any security policy implemented in a domain affects only those objects in that domain. Users with authority to perform tasks in one domain do not necessarily have authority to perform those tasks in other domains.

For small and moderately sized enterprises, one domain is usually sufficient. If only one domain is needed, no explicit action needs to be taken. Tivoli Access Manager automatically creates a domain called default, referred to as the management domain, as part of its initial configuration. This domain is used by Tivoli Access Manager to manage the security policy of all domains and is available for managing other protected resources as well.

In large enterprises, however, you might want to define two or more domains. Each domain is given a name and is established with a unique set of physical and logical resources. The security administrator can define the resources in a domain based on geography, business unit, or major organizational division within the enterprise. The security policy defined in the domain affects only those objects within the domain, which allows data to be partitioned and managed completely independently.
A multiple domain environment can be invaluable when there is a business need to keep a physical separation between different sets of data. Other benefits associated with multiple domains are as follows:

- **Increased security**
  Security policy data for each domain is mutually exclusive. Users, groups, and resources that are defined within a domain cannot be associated with another domain. For example, suppose that a user named John Doe is identified as JohnDoe in the Sales domain and as JDoe in the Advertising domain. Although the same person, each user ID is unique for each domain. Therefore, resources available to user JohnDoe can be granted access only by the unique identity he is defined by in that domain (Sales) or by groups that are defined in the Sales domain that JohnDoe is a member of. Likewise, user JDoe, even though it is the same person, can be granted access only by the unique identity he is defined by in the Advertising domain.

- **Simplified administration**
  You can assign independent administrators to handle policy management tasks for each domain. For example, assume that you are an IT specialist for a large corporation, assigned to deploy Tivoli Access Manager from a single data center. You could create a separate domain (with a unique policy database and an administrator) for each organization, division, or geographic area in your company. As users, groups, or resources change, the assigned administrator is responsible for updating the security policy for that particular domain. This domain administrator can also delegate administration tasks to others within that specific domain.

An administrator assigned to a specific domain has authority only within that domain. However, by default, an administrator can view users and groups defined in the user registry that are not necessarily Tivoli Access Manager users or groups. This is beneficial if, for example, an administrator wants to import a user or group from a different domain. Conversely, if you are the administrator of the management domain and want to limit the registry data that a domain administrator can access, you can add the allowed-registry-substrings stanza entry to the [domains] stanza in the policy server configuration file (ivmgrd.conf).

For more information about managing domains, see Chapter 5, "Domain management," on page 61.

---

**Protected object space**

Tivoli Access Manager represents resources within a domain using a virtual representation called the **protected object space**. The protected object space is the logical and hierarchical portrayal of resources belonging to a domain.

The protected object space consists of two types of objects:

- **Resource objects**
  Resource objects are the logical representation of actual physical resources, such as files, services, Web pages, message queues, and so on, in a domain.

- **Container objects**
  Container objects are structural components that allow you to group resource objects hierarchically into distinct functional regions.
Security policy can be applied to both types of objects. Figure 11 shows a logical representation of a protected object space with multiple container and resource objects.

Figure 11. Tivoli Access Manager protected object space

The structural top, or start, of the protected object space is the root container object, which is represented by a forward slash (/) character. Below the root container object are one or more container objects. Each container object represents an object space consisting of a related set of resources. These resources can be resource objects or other container objects.

Tivoli Access Manager creates an object space called /Management that consists of the objects used to manage Tivoli Access Manager itself. Each resource manager that protects a related set of resources creates its own object space. For instance, the WebSEAL component, which protects Web-based information and resources, creates an object space called /WebSEAL. These companion applications are referred to as blades. Figure 12 shows the /Management object space within the protected object space.

Figure 12. Regions of the Tivoli Access Manager protected object space
Users and groups

Tivoli Access Manager maintains information about Tivoli Access Manager users and groups in the user registry. Users and groups that already exist in the user registry can be imported into Tivoli Access Manager. If a user or group does not already exist in the user registry, it can be created directly within Tivoli Access Manager.

When a domain is created, a special user known as the *domain administrator* is created. For the management domain, the user ID of the domain administrator (sec_master) and the password are set during the configuration of the Tivoli Access Manager policy server. For other domains, the user ID and password of the domain administrator are established when the domain is created. The domain administrator has nearly complete control of the domain. Think of the domain administrator as the Tivoli Access Manager equivalent of the UNIX *root* account, or the Microsoft Windows *Administrator* user.

The domain administrator is added as a member of the Tivoli Access Manager *iv-admin* group within the domain. The *iv-admin* group represents those users with domain administration privileges. Care should be exercised when adding users to this group to ensure that you do not compromise the security of your domain.

Security policy

Access to objects within a domain is controlled by applying *security policy* (POPs, ACLs, and rules) to the container and resource objects in the protected object space. Security policy can be explicitly applied to an object or inherited by the object from objects above it in the hierarchy. You need to apply an explicit security policy in the protected object space only at those points in the hierarchy where the rules must change.

![Diagram](image)

*Figure 13. Explicit and inherited policies*

Security policy is defined using a combination of:

- Access control lists (ACLs)
  
  An access control list, or ACL, specifies what set of predefined actions a set of users and groups can perform on an object. For example, a specific set of groups or users can be granted read access to the object.
- Protected object policies (POPs)
A protected object policy, or POP, specifies access conditions associated with an object that affect all users and groups. For example, a time-of-day restriction can be placed on the object that excludes all users and groups from accessing the object during the specified time.

- Authorization rules
  An authorization rule specifies a complex condition that is evaluated to determine whether access is permitted. The data used to make this decision can be based on the context of the request, the current environment, or other external factors. For example, a request to modify an object more than 5 times in an 8 hour period could be denied.

Security policy is implemented by strategically applying ACLs, POPs, and authorization rules to those resources requiring protection. The Tivoli Access Manager authorization service makes decisions to permit or deny access to resources based on the credentials of the user making the request, and the specific permissions and conditions set in the ACLs, POPs, and authorization rules.

The authorization engine uses the following algorithm to process the policy attached to a protected object:

1. Check ACL permissions. See “Evaluating an ACL” on page 38 for information on the ACL evaluation process.
   The ACL is also checked to determine whether the user (for whom the authorization check is being made) has the additional privilege of being unaffected by POP or authorization rule policy. This privilege is bestowed when the user’s effective ACL for access to the object contains the B permission to denote that POP policy is ignored, or the R permission to denote that authorization rule policy is ignored.

2. When an authorization rule is attached to the object and the user does not have the privilege of being unaffected by authorization rules, verify that all of the ADI is present for the coming rule evaluation. If it is not, then find it by querying one of the available sources.

3. When there is a POP attached, check the Internet Protocol (IP) endpoint authentication method policy.

4. When there is a POP attached, check the time-of-day policy on the POP.

5. When there is a POP attached, check the audit-level policy on the POP, and audit the access decision as directed.

6. When an authorization rule is attached to the object and the user does not have the privilege of being unaffected by authorization rules, check the authorization rule policy.

7. When an external authorization service (EAS) operation or POP trigger applies to this access decision, invoke the external authorization services that apply.

If any of the ACL, POP, or authorization rule evaluations fail, the access request is denied. The external authorization service can override this decision on its own, if it has been designed to do so, or it might choose not to participate in the authorization decision at all.

Every ACL, POP, or authorization rule can be thought of as a policy. You fill in the policy, specifying the appropriate access conditions. After the policy is complete, you apply it to any number of resources within the domain. Subsequent changes to the policy are automatically reflected across the domain.
ACL policies

The policy that defines who has access to an object, and what operations can be performed on the object, is known as the ACL policy. Each ACL policy has a unique name and can be applied to multiple objects within a domain.

An ACL policy consists of one or more entries describing:
- The names of users and groups whose access to the object is explicitly controlled
- The specific operations permitted to each user, group, or role
- The specific operations permitted to the special any-other and unauthenticated user categories

Using ACL policies with the authorization service

Tivoli Access Manager relies on ACL policies to specify the conditions necessary for a particular user to perform an operation on a protected object. When an ACL is attached to an object, entries in the ACL specify what operations are allowed on this object and who can perform those operations.

Tivoli Access Manager uses a default set of actions that cover a wide range of operations. Actions, or permissions, are represented by single alphabetic ASCII characters (a-z, A-Z). Each permission is displayed (by pdadmin or the Web Portal Manager) with a label describing the operation it governs. In addition, the Web Portal Manager groups the ACLs according to their use in a particular part of the object space (such as WebSEAL) or their use across the entire object space (Base, Generic).

Resource manager software typically contains one or more operations that are performed on protected resources. Tivoli Access Manager requires these applications to make calls into the authorization service before the requested operation is allowed to progress. This call is made through the authorization application programming interface (authorization API) for both Tivoli Access Manager services and other applications.

The authorization service uses the information contained in the ACL to make a simple yes or no response to the question: Does this user (group) have the r permission (for example) to 'view' the requested object?

The authorization service has no knowledge about the operation requiring the r permission. It is merely noting the presence, or not, of the r permission in the ACL entry of the requesting user or group.

The authorization service is completely independent of the operations being requested. This is why it is easy to extend the benefits of the authorization service to other applications.

Evaluating an ACL

Tivoli Access Manager follows a specific evaluation process to determine the permissions granted to a particular user by an ACL. When you understand this process, you can determine how best to keep unwanted users from gaining access to resources.

Evaluating authenticated requests

Tivoli Access Manager evaluates an authenticated user request in the following order:
1. Match the user ID with the ACL’s user entries. The permissions granted are those in the matching entry.
   
   *Successful match: evaluation stops here. Unsuccessful match: continue to the next step.*

2. Determine the groups to which the user belongs and match with the ACL’s group entries:
   
   If more than one group entry is matched, the resulting permissions are a logical “or” (most permissive) of the permissions granted by each matching entry.

   *Successful match: evaluation stops here. Unsuccessful match: continue to the next step.*

3. Grant the permissions of the **any-other** entry (if it exists).

   *Successful match: evaluation stops here. Unsuccessful match: continue to the next step.*

4. An implicit **any-other** entity exists when there is no **any-other** ACL entry. This implicit entry grants no permissions.

   *Successful match: no permissions granted. End of evaluation process.*

### Evaluating unauthenticated requests

Tivoli Access Manager evaluates an unauthenticated user by granting the permissions from the ACL’s **unauthenticated** entry.

The **unauthenticated** entry is a mask (a bitwise “and” operation) against the **any-other** entry when permissions are determined. A permission for **unauthenticated** is granted only if the permission also appears in the **any-other** entry.

Because **unauthenticated** depends on **any-other**, it makes little sense for an ACL to contain **unauthenticated** without **any-other**. If an ACL does contain **unauthenticated** without **any-other**, the default response is to grant no permissions to **unauthenticated**.

### Protected object policies

A protected object policy (POP) specifies security policy that applies to an object regardless of what user or what operation is being performed. Each POP has a unique name and can be applied to multiple objects within a domain.

The purpose of a POP is to impose access conditions on an object based on the time of the access and to indicate whether the access request should be audited. Specifically, the conditions you can apply are:

- POP attributes, such as warning mode, audit level, and time-of-day.
  
  More details about these attributes are in “Configuring the POP attributes” on page 93.

- Authentication strength POP (step-up).
  
  More details about this policy are in “Authentication strength POP policy (step-up)” on page 97.

- Quality of protection POP.
  
  More details about this policy are in “Quality of protection POP policy” on page 99.

- Network-based authentication POP.
  
  More details about this policy are in “Network-based authorization POP policy” on page 95.
Authorization rules

An authorization rule policy specifies security policy that applies to an object based on a variety of conditions, such as context and environment. Each authorization rule policy has a unique name and can be applied to multiple objects within a domain.

Like ACLs and POPs, authorization rules are defined to specify conditions that must be met before access to a protected object is permitted. An authorization rule is created using a number of conditions that are based on data supplied to the authorization engine within the user credential, from the resource manager application or from the encompassing business environment. These conditions are evaluated as a Boolean expression to determine if access to the object should be granted or denied.

The language of an authorization rule allows you to work with complex, structured data. You can examine values in the rule data and make informed access decisions. The data used in an access decision can be defined statically within the system or defined during the course of a business process. Rules give you the flexibility of the policy defined by an external authorization service without requiring that you develop and build the logic of an external authorization service into a shared library plug-in.

How authorization rules differ from ACLs and POPs

ACLs take a given predefined set of operations and control which users and groups have permission to perform those operations on a protected object. For example, a user’s ability to read data associated with an object is either granted or denied by an ACL policy. POPs apply to all users and groups and control conditions that are specific to a particular protected object. For example, time-of-day access excludes all users and groups from accessing an object outside of the times set in the time-of-day policy.

Rules allow you to make decisions based on the attributes of a person or object and the context and environment surrounding the access decision. For example, you can use a rule to implement a time-of-day policy that depends on the user or group. You also can use a rule to extend the access control capabilities that ACLs provide by implementing a more advanced policy, such as one based on quotas. While an ACL can grant a group permission to write to a resource, a rule can go a step further by allowing you to determine if a group has exceeded a specific quota for a given week before permitting that group to write to a resource.

When to use authorization rules

In the Tivoli Access Manager authorization process, all three policy objects—the ACL, the POP, and the authorization rule—must permit access to a protected object before access to the object is granted. Authorization rules provide the flexibility needed to extend an ACL or POP by tailoring security policy to your needs.

Although authorization rules can be used to extend the policy implemented by other Tivoli Access Manager policy types, they are not simply extensions of the existing policy types. An authorization rule is a policy type that is rich enough in functionality to replace the ACL and POP. However, using ACLs and POPs generally provides better performance. Therefore, use a rule to complement these policies instead of replacing them.
Guidelines for a secure object space

The following guidelines are applicable for a secure object space:

- Set high-level security policy on container objects at the top of the object space. Set exceptions to this policy with explicit ACLs, POPs, and authorization rules on objects that are lower in the hierarchy.

- Arrange your protected object space so that most objects are protected by inherited, rather than explicit, ACLs, POPs, and authorization rules.
  Reduce the risk of an error that could compromise your network by simplifying the maintenance of your tree. Inherited security policy lowers maintenance because it reduces the number of ACLs, POPs, and authorization rules that you must maintain.

- Position new objects in the tree where they inherit the appropriate permissions.
  Arrange your object tree into a set of subtrees, where each subtree is governed by a specific access policy. You determine the access policy for an entire subtree by setting explicit ACLs, POPs, and authorization rules at the root of the subtree.

- Create a core set of ACLs, POPs, and authorization rules, and reuse these policies wherever necessary.
  Because ACL, POP, and authorization rule policies are a single source definition, any modifications to the policy impacts all objects associated with the ACL, POP, or authorization rule.

- Control user access through the use of groups.
  It is possible for an ACL to consist of only group entries. Individual user entries are not required in the ACL when the users can be categorized into groups instead. Authorization rules can also be written to consider an individual’s group memberships rather than the individual specifically. This can reduce the complexity of the rule logic considerably.
  Access to an object by individual users can be efficiently controlled by adding users to or removing users from these groups.
Chapter 4. Default security policy

Tivoli Access Manager establishes a default security policy to protect all objects in a domain. A set of administrative users and groups is established and granted a predefined set of permissions. The default security policy is described in this chapter.

Default administration users and groups

At installation, Tivoli Access Manager provides several important administration groups. By default, these users and groups are given special permissions to control and manage all operations in a domain. (This default security policy is defined by the access control lists (ACLs) created during configuration.)

The following sections detail the specific roles assigned to each of these users and groups at installation time, and explain how to create administration users.

group iv-admin

This group represents the administrator group. All members of this group are considered administrators of the domain by the default policy.

You can easily place users into an administration role by adding them to the iv-admin group. The danger with this procedure is that as soon as a user becomes a member of this group (with the default ACLs), that user has full rights to perform administration operations on any object in the entire namespace within the default policy.

When the policy server is configured, the administrator (sec_master) user is created and added to the iv-admin group. It is the combination of group memberships that grants sec_master complete rights for all operations within the management domain but only within the default policy. The sec_master user does not have rights to new groups created outside of the default policy unless it is added as a user or a member of a group.

user sec_master

The sec_master user is created when Tivoli Access Manager is initially installed and configured. The default policy makes the sec_master user a member of the iv-admin group, permitting it to perform all actions within Tivoli Access Manager. Think of this account as the equivalent of the UNIX root account, or the Microsoft Windows Administrator account.

group ivmgrp-servers

This group contains the policy servers and the policy proxy servers. By default, members of this group are authorized to delegate requests to other Tivoli Access Manager servers on behalf of the requestor.

Administration users

You can create administration accounts with varying degrees of responsibility. Responsibility is delegated to administrators through strategically placed administration ACLs.
The following list illustrates possible administration roles:

- **Security Policy Administrator**

  The security policy administrator is responsible for defining and organizing security policy within a domain. The administrator needs to be able to create, modify, and delete security policy. The traverse (T), browse (b), view (v), modify (m) and delete (d) permissions are required on the /Management/ACL, /Management/POP, and /Management/Rule resources.

  The administrator also needs the traverse (T), browse (b), and view (v) permissions to navigate the subtree of protected resources for which the administrator is responsible. In addition, the administrator needs the ability to attach (a) and detach (d) security policy to the same subtree. The administrator should also have the bypass POP (B) and bypass Rule (R) permissions so as not to be affected by security policies that apply to all users for the same subtree.

- **Protected Resource Administrator**

  The protected resource administrator is responsible for providing and removing a user’s access to one or more protected resources. These tasks entail adding and removing users from groups that have been defined in the security policy to have them, or adding and removing permissions with regards to the resource.

  The administrator must have traverse (T), browse (b), view (v), and add (A) permissions on the /Management/Groups protected resource or on the individual group that is defined in the /Management/Groups subtree.

- **Deployment Administrator**

  The deployment administrator is responsible for installation and configuration of the resource managers in the domain.

  The administrator must have the traverse (T), browse (b), view (v), modify (m), and delete (d) permissions on the /Management/Server protected resource. These permissions give the ability to configure resource managers into and out of the domain as well as update their configuration.

**Example administration ACL policies**

The following example illustrates how a user gains administration rights.

The following ACL on /WebSEAL gives administration rights to the user *adam*:

<table>
<thead>
<tr>
<th>user</th>
<th>permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>user sec_master</td>
<td>abcTdm1rx</td>
</tr>
<tr>
<td>group iv-admin</td>
<td>abcTdm1rx</td>
</tr>
<tr>
<td>group webseal-servers</td>
<td>gTdm1rx</td>
</tr>
<tr>
<td>group ivmgrd-servers</td>
<td>T1</td>
</tr>
<tr>
<td><strong>user adam</strong></td>
<td>abcTdm1rx</td>
</tr>
<tr>
<td>any-other</td>
<td>Trx</td>
</tr>
<tr>
<td>unauthenticated</td>
<td>Trx</td>
</tr>
</tbody>
</table>

**Defining and applying security policy**

Security administrators protect system resources by defining a security policy. Security policy consists of the access control lists (ACLs), protected object policies (POPs), and authorization rule that are applied to the object representations of the system resources to be protected in the object space. You can apply ACLs, POPs, and authorization rules to the same object.
The authorization service performs authorization decisions based on the policies applied to these objects. When a requested operation on a protected object is permitted, the resource manager responsible for the resource implements this operation.

One policy can dictate the protection parameters of many objects. Any change to an ACL, POP, or authorization rule affects all objects to which the policy is attached.

**Access control list**

An access control list (ACL) policy, or ACL policy, is the set of controls (permissions) that specifies the conditions necessary to perform certain operations on that resource. ACL policy definitions are important components of the security policy established for the domain. ACL policies, like all policies, are used to stamp an organization’s security standards onto the resources represented in the protected object space.

An ACL policy specifically controls the following:

- What operations can be performed on the resource
- Who can perform these operations

An ACL policy is made up of one or more entries that include user and group designations and either their specific permissions or rights.

![ACL policy](image)

*Figure 14. ACL policy*

**Default administration ACL policies**

The following default administration ACL policies are suggested starting points for securing management operations within a domain.

You can add entries for users, groups, any-other (any-authenticated), and unauthenticated to provide a broader range of control and better meet the requirements of your protected object space.

Note the users and groups in each ACL that contain the control (c) permission. Users and groups with the control permission own the ACL and have the power to modify the ACL entries.

A detailed description of permissions can be found in [“Default Tivoli Access Manager permissions (actions)” on page 75](#).

**Default root ACL policy**

Users and permissions for the default root ACL, default-root, include:
The root ACL is very basic—everyone can traverse the object space, but cannot perform any other actions. Typically, you would not need to change this. However, one useful function of the root ACL is to quickly deny access to the entire object space for an individual user or group.

Consider the following entry in the root ACL:

user john -----------------  

The consequence of this entry (no permissions) is that user john cannot even traverse the root container object. This user cannot gain access at all to the protected object space—regardless of any permissions granted lower down in the tree.

**Default /Management ACL policy**

Users and permissions for the Management ACL, default-management, include:

Group iv-admin TcmdbvaBR
Group ivmgrd-servers Ts
Any-other T

At installation, this ACL is attached to the /Management container object in the object space.

**Default /Replica ACL policy**

Users and permissions for the Replica management ACL, default-replica, include:

Group iv-admin TcbvaBR
Group ivmgrd-servers m
Group secmgrd-servers mdv
Group ivacld-servers mdv

**Default /Config ACL policy**

Users and permissions for the Config management ACL, default-config, include:

Group iv-admin TcmdbvaBR
Any-other T
Unauthenticated T

**Default /GSO ACL policy**

Users and permissions for the GSO management ACL, default-gso, include:

Group iv-admin TcmdbvaBNR
Any-other T
Unauthenticated T

**Default /Policy ACL policy**

Users and permissions for the Policy management ACL, default-policy, include:

Group iv-admin TcmdbvaBNR
Any-other T
Unauthenticated T

**Default /Domain ACL policy**

Users and permissions for the Domain management ACL, default-domain, include:

Group iv-admin TcmdbvaBNR
Group ivmgrd-servers v

**Default /Proxy ACL policy**

Users and permissions for the Proxy management ACL, default-management-proxy, include:
Management permissions

The Management region of the protected object space contains several submanagement container objects that require specific sets of permissions:

- “/Management/ACL permissions” on page 47
- “/Management/Action permissions” on page 48
- “/Management/POP permissions” on page 49
- “/Management/Server permissions” on page 49
- “/Management/Config permissions” on page 50
- “/Management/Policy permissions” on page 50
- “/Management/Replica permissions” on page 50
- “/Management/Users permissions” on page 51
- “/Management/Groups permissions” on page 52
- “/Management/GSO permissions” on page 53
- “/Management/Rule permissions” on page 53
- “/Management/Domain permissions” on page 54
- “/Management/Proxy permissions” on page 54

The following security considerations apply for the /Management region of the protected object space:

- The Management object begins the chain of ACL inheritance for the entire Management region of the object space.
- If you do not apply any other explicit ACLs, this object defines (through inheritance) the security policy for the entire Management object space.
- The traverse permission is required for access to /Management.

/Management/ACL permissions

This object allows administration users to perform high-level ACL management tasks that can impact the security policy for the domain.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>d delete</td>
<td>Delete an existing ACL policy. pdadmin acl delete</td>
</tr>
<tr>
<td>m modify</td>
<td>Create a new ACL policy. pdadmin acl create</td>
</tr>
<tr>
<td>v view</td>
<td>List and find view ACLs; show ACL details. This permission must be in an entry of an ACL attached to /Management/ACL. pdadmin acl find pdadmin acl list pdadmin acl show</td>
</tr>
</tbody>
</table>

The acl find command shows the list of protected resources where this ACL is attached. You must have the view (v) permission on those protected resources before they can be shown.
You must create ACL administrator entries in the effective ACL policy for the /Management/ACL object. The administrator’s ACL entry can contain any of the above permissions. These permissions give the administrator powers to create, view, and delete ACL policies.

An ACL administrator cannot modify an existing ACL unless there is an entry in that ACL for the administrator containing the control (c) permission. Only the owner of an ACL can modify its entries.

Note that the creator of a new ACL policy (m on /Management/ACL) becomes the first entry in that ACL—with the TcmdbsvaBIR permissions set by default.

For example, if sec_master is an administrator entry in the default-management ACL, with m permission, sec_master can create a new ACL, with TcmdbsvaBIR permissions.

Ownership of the default-management ACL itself is given to the group iv-admin by default.

/Management/Action permissions
This object allows administration users to manage custom actions and action groups. Action tasks and associated permissions include:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>delete</td>
</tr>
<tr>
<td></td>
<td>pdadmin action delete</td>
</tr>
<tr>
<td></td>
<td>pdadmin action group delete</td>
</tr>
<tr>
<td>m</td>
<td>modify</td>
</tr>
<tr>
<td></td>
<td>Create a new action or action group.</td>
</tr>
<tr>
<td></td>
<td>pdadmin action create</td>
</tr>
<tr>
<td></td>
<td>pdadmin action group create</td>
</tr>
</tbody>
</table>

Note: The following commands do not require special permissions:


Tivoli Access Manager provides authorization services to resource managers. For example, resource managers that are part of the Tivoli Access Manager family include WebSEAL (for Web resource managers) and Tivoli Access Manager for Business Integration (for messaging applications).

Resource managers can make calls to the authorizations service through the authorization API. Three necessary steps required to integrate a resource manager with the authorization service include:

- Define the resource manager’s object space
- Define the resource manager’s action groups and actions
- Apply permissions on resources (objects) needing protection

The administrator of an resource manager’s object space can use the pdadmin utility to define new permissions and actions. Resource managers generally define the actions and action groups applicable to the resources that they are protecting.
The administrator must have the m and d Management/Action permissions to create and delete these permissions or actions.

/Management/POP permissions
This object allows administration users to manage protected object policies. All permissions must appear in entries for ACLs on /Management/POP. Action tasks and associated permissions include:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>delete</td>
</tr>
<tr>
<td>m</td>
<td>modify</td>
</tr>
<tr>
<td>v</td>
<td>view</td>
</tr>
<tr>
<td>B</td>
<td>Bypass POP</td>
</tr>
</tbody>
</table>

The pop find command shows the list of protected resources where this ACL is attached. You must have the view (v) permission on those protected resources before they can be shown.

/Management/Server permissions
The /Management/Server container object of the protected object space allows administrators to perform server management tasks (when appropriate permissions are set).

Server management controls are used to determine whether a user has permission to view configured resource managers, initiate a replication to one more resource managers, and to enable runtime tracing features on behalf of resource managers.

Resource managers become available in the list of resource managers after they have been configured into the domain. Resource managers are removed when they are unconfigured.

The viewable resource manager information contains information that allows other Tivoli Access Manager servers, particularly the policy server (pdmgrd), to locate and communicate with that server.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>server</td>
</tr>
<tr>
<td>v</td>
<td>view</td>
</tr>
<tr>
<td>t</td>
<td>trace</td>
</tr>
</tbody>
</table>
**/Management/Config permissions**  
The /Management/Config container object of the protected object space allows administrators to perform configuration management tasks (when appropriate permissions are set).

Configuration management controls are used to determine whether a user has permission to configure, unconfigure, or update the configuration of a resource manager.

A server definition is created for a particular resource manager (such as WebSEAL) or the authorization server (`pdacld`) as part of the configuration process. The definition for a server is also deleted when the server is unconfigured.

Server definitions contain information that allows other Tivoli Access Manager servers, particularly the policy server (`pdmgrd`), to locate and communicate with that server.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>modify</td>
</tr>
<tr>
<td></td>
<td>Configuration into a domain.</td>
</tr>
<tr>
<td></td>
<td><code>svrsslcfg -config</code></td>
</tr>
<tr>
<td></td>
<td><code>svrsslcfg -modify</code></td>
</tr>
<tr>
<td>d</td>
<td>delete</td>
</tr>
<tr>
<td></td>
<td>Unconfiguration.</td>
</tr>
<tr>
<td></td>
<td><code>svrsslcfg -unconfig</code></td>
</tr>
</tbody>
</table>

**/Management/Policy permissions**  
The /Management/Policy container object of the protected object space allows administrators to authorize the `policy get` and `policy set` commands (when appropriate permissions are set).

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>view</td>
</tr>
<tr>
<td></td>
<td>Required for <code>policy get</code> operations.</td>
</tr>
<tr>
<td>m</td>
<td>modify</td>
</tr>
<tr>
<td></td>
<td>Required for <code>policy set</code> operations.</td>
</tr>
</tbody>
</table>

**/Management/Replica permissions**  
The /Management/Replica container object of the protected object space controls the replication of the authorization database. High-level controls on this object affect the operation of the policy server and the security managers in the domain.

Replica management controls are used to determine which resource managers are allowed to download the master authorization policy database to their local file system.

Controls and associated permissions include:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>view</td>
</tr>
<tr>
<td></td>
<td>Read the master authorization database.</td>
</tr>
</tbody>
</table>

All Tivoli Access Manager servers that maintain a local replica of the authorization database, which includes all resource managers and authorization servers, must be granted view (v) permission on the /Management/Replica object. The replication process requires that these processes be allowed to view and access entries out of the master authorization policy database.
The Tivoli Access Manager installation automatically grants read permission to any server requiring access to the authorization policy database. When a resource manager configures into the domain, it is automatically added as a member to the group `ivacld-servers`. This group, by default, is given permission to download the master authorization policy database.

/Management/Users permissions
This object allows administration users to manage user accounts. Action tasks and associated permissions include:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>Delete a user account.</td>
</tr>
<tr>
<td></td>
<td><code>pdadmin user delete</code></td>
</tr>
<tr>
<td>m</td>
<td>Modify user account details.</td>
</tr>
<tr>
<td></td>
<td><code>pdadmin user modify authentication-mechanism</code></td>
</tr>
<tr>
<td></td>
<td><code>pdadmin user modify account-valid</code></td>
</tr>
<tr>
<td></td>
<td><code>pdadmin user modify gsouser</code></td>
</tr>
<tr>
<td></td>
<td><code>pdadmin user modify description</code></td>
</tr>
<tr>
<td>N</td>
<td>Create a new user and optionally assign that user to one or more groups. Import group data from the user registry.</td>
</tr>
<tr>
<td></td>
<td><code>pdadmin user create</code></td>
</tr>
<tr>
<td></td>
<td><code>pdadmin user import</code></td>
</tr>
<tr>
<td>v</td>
<td>List user accounts and show user account details.</td>
</tr>
<tr>
<td></td>
<td><code>pdadmin user list</code></td>
</tr>
<tr>
<td></td>
<td><code>pdadmin user list-dn</code></td>
</tr>
<tr>
<td></td>
<td><code>pdadmin user list-gsouser</code></td>
</tr>
<tr>
<td></td>
<td><code>pdadmin user show</code></td>
</tr>
<tr>
<td></td>
<td><code>pdadmin user show-dn</code></td>
</tr>
<tr>
<td></td>
<td><code>pdadmin user show-groups</code></td>
</tr>
<tr>
<td>W</td>
<td>Reset and validate a user password.</td>
</tr>
<tr>
<td></td>
<td><code>pdadmin user modify password</code></td>
</tr>
<tr>
<td></td>
<td><code>pdadmin user modify password-valid</code></td>
</tr>
</tbody>
</table>

The password (W) permission allows password resets and is appropriate to give to help desk administrators so they can assist users who have forgotten their passwords. This permission allows an administrator to reset the forgotten password and then to use the `user modify password-valid` command to set a value of no. This action allows the user to log on and then forces the user to immediately apply a new password. Note that setting `user modify password-valid` to no for a user does not indicate if the password is invalid due to the `max-password-age` policy, which is a global setting. The policy `set max-password-age` command sets the maximum time before a password expires.

The ability for an administrator to manage all user accounts is controlled by permissions on the /Management/Users object. For example, if an administrator has view (v) permission on the /Management/Users object, that administrator is able to view information about all users.

To limit the scope of an administrator’s control to a specific group of users, remove the administrator’s permissions from the /Management/Users object and apply permissions to the /Management/Groups object associated with the group to be managed instead. For example, if an administrator is given view (v) permission on the /Management/Groups/Accounting object, that administrator is only able to view information about users in the Accounting group.
If an administrator has view (v) permission to any group that the user is a member of, the administrator is able to view that user’s information. Adding view (v) permission to the /Management/Groups object itself allows an administrator to view information about any user who is a member of any group.

Access granted by the /Management/Users object overrides any access restrictions imposed by “delegated administration” policy ACLs under /Management/Groups/group_name. For information on delegated administration, see Chapter 16, “Delegated administration,” on page 155.

/Management/Groups permissions
This object allows administration users to manage groups and group membership. Action tasks and associated permissions include:

<table>
<thead>
<tr>
<th>Permission</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
</table>
| d          | delete    | Delete a group.  
|            |           | pdadmin group delete |
| m          | modify    | Modify group descriptions. Remove one or more user members of a group.  
|            |           | pdadmin group modify description  
|            |           | pdadmin group modify remove |
| N          | create    | Create a new group. Import group data from the user registry.  
|            |           | pdadmin group create  
|            |           | pdadmin group import |
| v          | view      | List groups and show group details.  
|            |           | pdadmin group list  
|            |           | pdadmin group list-dn  
|            |           | pdadmin group show  
|            |           | pdadmin group show-dn  
|            |           | pdadmin group show-members |
| A          | add       | Add one or more users to a group.  
|            |           | pdadmin group modify add |

The add (A) permission is required on your entry in the ACL on a group to allow you to add existing users to your group. Use the `pdadmin user create` command (which requires the N permission) to create new users and optionally place them in one or more existing groups.

The capability of adding existing users to your group is powerful because the owner of a group has control over all user members of the group. If you, as the owner of the group, also have the delete (d) permission, you can delete this user from the entire domain.

The ability for an administrator to manage all groups is controlled by permissions on the /Management/Groups object. For example, if an administrator has delete (d) permission on the /Management/Groups object, that administrator is able to delete any group.

To limit the scope of an administrator’s control to a specific group, apply permissions to the object associated with the group instead. For example, if an administrator is given delete (d) permission on the /Management/Groups/Travel/Europe object, that administrator is able to delete the
Similarly, if the administrator is given the delete (d) permission on the /Management/Groups/Travel container object, that administrator is able to delete any group within the container object.

Permissions on /Management/Groups objects also affect an administrator’s ability to manage users who are part of those groups. Thus, giving an administrator delete (d) permission on a group allows the administrator to delete a user that is a member of the group. Similarly, if an administrator has view (v) permission on a group, information on users associated with those groups can be viewed.

/Management/GSO permissions
The /Management/GSO container object of the protected object space allows administrators to perform Global Sign-On (GSO) management tasks (when appropriate permissions are set).

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>create</td>
</tr>
<tr>
<td></td>
<td>pdadmin rsrcc create</td>
</tr>
<tr>
<td></td>
<td>pdadmin rsrccgroup create</td>
</tr>
<tr>
<td></td>
<td>pdadmin rsrccred create</td>
</tr>
<tr>
<td></td>
<td>(All the above commands also require m)</td>
</tr>
<tr>
<td>d</td>
<td>delete</td>
</tr>
<tr>
<td></td>
<td>pdadmin rsrcc delete</td>
</tr>
<tr>
<td></td>
<td>pdadmin rsrccgroup delete</td>
</tr>
<tr>
<td></td>
<td>pdadmin rsrccred delete</td>
</tr>
<tr>
<td></td>
<td>(All the above commands also require m)</td>
</tr>
<tr>
<td>m</td>
<td>modify</td>
</tr>
<tr>
<td></td>
<td>pdadmin rsrccgroup modify</td>
</tr>
<tr>
<td></td>
<td>pdadmin rsrccred modify</td>
</tr>
<tr>
<td>v</td>
<td>view</td>
</tr>
<tr>
<td></td>
<td>pdadmin rsrcc list</td>
</tr>
<tr>
<td></td>
<td>pdadmin rsrccgroup list</td>
</tr>
<tr>
<td></td>
<td>pdadmin rsrccred list</td>
</tr>
<tr>
<td></td>
<td>pdadmin rsrcc show</td>
</tr>
<tr>
<td></td>
<td>pdadmin rsrccgroup show</td>
</tr>
<tr>
<td></td>
<td>pdadmin rsrccred show</td>
</tr>
</tbody>
</table>

/Management/Rule permissions
This object allows administration users to manage authorization rule policies. All permissions must appear in entries for ACLs on /Management/Rule. Action tasks and associated permissions include:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>Bypass Rule</td>
</tr>
<tr>
<td></td>
<td>Override the authorization rule policy on an object.</td>
</tr>
<tr>
<td>d</td>
<td>delete</td>
</tr>
<tr>
<td></td>
<td>Delete an authorization rule.</td>
</tr>
<tr>
<td></td>
<td>pdadmin authzrule delete</td>
</tr>
<tr>
<td>m</td>
<td>modify</td>
</tr>
<tr>
<td></td>
<td>Create authorization rules and modify authorization rule attributes.</td>
</tr>
<tr>
<td></td>
<td>pdadmin authzrule create</td>
</tr>
<tr>
<td></td>
<td>pdadmin authzrule modify</td>
</tr>
<tr>
<td>v</td>
<td>view</td>
</tr>
<tr>
<td></td>
<td>Find and list authorization rules and show authorization rule details.</td>
</tr>
<tr>
<td></td>
<td>pdadmin authzrule find</td>
</tr>
<tr>
<td></td>
<td>pdadmin authzrule list</td>
</tr>
<tr>
<td></td>
<td>pdadmin authzrule show</td>
</tr>
</tbody>
</table>
The authzrule find command shows the list of protected resources where this rule is attached. You must have the view (v) permission on those protected resources before they can be shown.

/Management/Domains permissions
The /Management/Domains container object of the protected object space allows administrators to perform domain management tasks (when appropriate permissions are set).

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>modify</td>
</tr>
<tr>
<td></td>
<td>pdadmin domain create</td>
</tr>
<tr>
<td></td>
<td>pdadmin domain modify</td>
</tr>
<tr>
<td>v</td>
<td>view</td>
</tr>
<tr>
<td></td>
<td>pdadmin domain list</td>
</tr>
<tr>
<td></td>
<td>pdadmin domain show</td>
</tr>
<tr>
<td>d</td>
<td>delete</td>
</tr>
<tr>
<td></td>
<td>pdadmin domain delete</td>
</tr>
</tbody>
</table>

/Management/Proxy permissions
The /Management/Proxy container object of the protected object space allows administrators or resource managers to perform delegated management tasks (when appropriate permissions are set).

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td>delegate</td>
</tr>
<tr>
<td></td>
<td>Allows administrators and resource managers to act on the behalf of the specified credential.</td>
</tr>
</tbody>
</table>

Sparse security policy model
To secure network resources in a protected object space, each object must be protected by security policy.

You can assign security policy to an object in one of two ways:
- Attach an explicit security policy on the object.
- Allow the object to inherit its security policy from a preceding container object in the hierarchy.

Adopting an inherited security scheme can greatly reduce the administration tasks for a domain. This section discusses the concepts of inherited, or sparse security policies.

Security policy inheritance
The power of security policy inheritance is based on this principle: Any object without an explicitly attached security policy inherits the policy of its nearest container object with an explicitly set security policy. In other words, all objects without explicitly attached security policies inherit security policy from container objects with explicitly attached security policies. A particular chain of inheritance is broken when you attach an explicit security policy on an object.

Security policy inheritance simplifies the task of setting and maintaining access controls on a large protected object space. In a typical object space, you need to attach only a few ACLs at key locations to secure the entire object space; hence, it is called a sparse security policy model.
A typical object space begins with a single explicit security policy attached to the root container object. The root ACL must always exist and can never be removed. Normally, this is an ACL with very little restriction. All objects located in the object space inherit this ACL.

When a region or subtree in the object space requires different access control restrictions, you attach an explicit security policy at the root of that subtree. This interrupts the flow of inherited security policies from the primary object space root to that subtree. A new chain of inheritance begins from this newly created explicit security policy.

**Default root ACL policy**

Tivoli Access Manager checks inheritance beginning with the root of the protected object space. If you do not explicitly set ACLs on any other objects in the tree, the entire tree inherits this root ACL.

There is always an explicit ACL policy set at the root of the protected object space. An administrator can replace this ACL with another ACL containing different entries and permission settings. But the root ACL can never be completely removed.

The root ACL policy is explicitly set during the initial Tivoli Access Manager installation and configuration.

Users and permissions for the default root ACL — default-root — include:

<table>
<thead>
<tr>
<th>Group</th>
<th>Control permission (c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>iv-admin</td>
<td>T</td>
</tr>
<tr>
<td>Any-other</td>
<td>T</td>
</tr>
<tr>
<td>Unauthenticated</td>
<td>T</td>
</tr>
</tbody>
</table>

**Control permission (c)**

The control permission is a powerful permission that gives you ownership of an ACL policy. Control allows you to modify the entries in the ACL. This means that you have the power to create entries, delete entries, grant permissions, and take away permissions.

The administrator who wants to delete an ACL from the list of ACL policies must have an entry in that ACL and must have the control permission set in that entry.

The control permission allows you to grant administration powers to another user, such as the ability to attach (a) that ACL to objects. You must use the control permission with great care because of its powerful ownership properties.

**Traverse permission (T)**

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 applied only to container objects in the protected object space. The traverse permission specifies that a user, group, any-other, or unauthenticated 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.
Note that, if there are no permissions defined for a user, the user cannot even traverse the root container object. This user cannot gain access at all to the protected object space, regardless of any permissions that might be granted lower down in the tree.

A protected object is accessible 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. Figure 15 illustrates how the traverse permission works. Within the ACME Corporation, there is an Engineering container object (directory), which also contains a TechPubs container object (subdirectory). User kate, a member of the Sales department, requires traversing to the Engineering/TechPubs directory to review a release note file. The administrator provides traverse for any-other at the root. The administrator provides traverse for group sales on the Engineering directory. The TechPubs directory inherits the ACL from the Engineering directory. Although Kate has no other permissions in these two directories, she can pass (traverse) through these directories in order to access the release_note file. Because this file has read permission for user kate, she can view the file.

Figure 15. Traverse permission

You can easily restrict access to the hierarchy below a given container object without resetting individual permissions on these objects. Simply remove the traverse permission from the appropriate ACL entry. Removing traverse permission on a directory object protects all objects lower in the hierarchy, even if those objects contain other less restrictive ACLs.

For example, if group sales did not have the traverse permission on the Engineering directory, Kate could not access the release note file, even though she has read permission for the file.

Resolving an access request
Inheritance begins with the root ACL and impacts all objects in the object space until it reaches an object with an explicit ACL. At this point, a new chain of inheritance begins.
Objects below an explicitly set ACL inherit the new access controls. If you delete an explicit ACL, access control for all objects reverts back to the nearest directory or container object with an explicitly set ACL.

When a user tries to access a secure object (such as a Web document), Tivoli Access Manager checks whether the user has the permissions to access the object. It does this by checking every object along the object hierarchy for the proper inherited or explicitly set permissions.

A user is denied access to an object if any directory or container object in the hierarchy above does not include the traverse permission for that user. Access is also denied if the target object does not contain sufficient permissions to perform the requested operation.

In order to succeed an access check, the requestor must have both of the following:
- Permission to traverse the path to the requested object.
- Appropriate permissions on the requested object.

The process of resolving whether a user can read (view) an object is as shown: /acme/engineering/project_Y/current/report.html

Tivoli Access Manager checks for the following:
- Traverse permission on the explicitly set root ACL (/).
- Traverse permission on any explicit ACLs attached to the directories: acme, engineering, project_Y, and current.
- Read permission on the file itself (report.html).

The user is denied access if the user fails the access check at any of these points along the object hierarchy.

**Applying ACL policies to different object types**

Permissions for a variety of operations can be set in an ACL policy. Only a subset of these possible operations might be relevant for a specific object to which the ACL is attached.

The reason for this behavior is related to the two features of Tivoli Access Manager that are designed to make administration easier:
- ACL policies
- ACL inheritance

ACL policies allow you to attach the same ACL definition to multiple objects in the protected object space. The ACL definition consists of enough entries to meet the requirements of all objects to which the ACL is applied; however, each individual object might be affected by only a few of the entries.

In the ACL inheritance model, any object without an attached explicit ACL policy “inherits” the policy definitions from the nearest ACL applied to an object above it in the hierarchy.

In summary, an ACL policy has to describe the necessary permissions for all object types that it is applied to — and not just the object that it is attached to.

**ACL policy inheritance example**

Figure 16 on page 58 illustrates the impact of a mixture of inherited and explicit ACLs in a corporate object space.
A corporate object space has a general security policy set at the root object. Root is followed by the /WebSEAL container object and individually controlled departmental sub-trees.

In this example, the sales group is given ownership of their departmental subtree. Note that the ACL on this subtree no longer acknowledges the unauthenticated or any-other entry types.

The year-to-date sales file (ytd.html) has an explicit ACL that grants read permission to members of the sales-vp group (who are also members of the sales group).

Note: This ACL scheme need not be changed with the addition or subtraction of users within the domain. New users are simply added to the appropriate groups. Likewise, users can be removed from those groups.

Figure 16. ACL inheritance example

Protected object policies

ACL policies provide the authorization service with information that results in a yes or no answer on a request to access a protected object and perform some operation on that object.

In contrast to ACL policies, protected object policies (POPs) contain additional conditions on the request that are passed back to Tivoli Access Manager Base and the resource manager along with the yes ACL policy decision from the authorization service. It is the responsibility of Tivoli Access Manager and the resource manager to enforce the POP conditions.
Available attributes for a POP are listed as follows:

<table>
<thead>
<tr>
<th>Enforced by Tivoli Access Manager Base</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POP attribute</td>
<td>Description</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the policy. This relates to the pop-name variable in the pdadmin pop online commands.</td>
</tr>
<tr>
<td>Description</td>
<td>Descriptive text for the policy. This appears in the pdadmin pop show command.</td>
</tr>
<tr>
<td>Warning mode</td>
<td>Provides administrators a means to test ACLs, POPs, and authorization rules.</td>
</tr>
<tr>
<td>Audit level</td>
<td>Specifies the type of auditing: all, none, successful access, denied access, errors.</td>
</tr>
<tr>
<td>Time-of-day Access</td>
<td>Day and time restrictions for successful access to the protected object.</td>
</tr>
<tr>
<td>IP endpoint authorization method policy</td>
<td>Specifies authorization requirements for access from members of external networks.</td>
</tr>
<tr>
<td>EAS trigger attributes</td>
<td>Specifies an External Authorization Service (EAS) plug-in that will be invoked to make an authorization decision using the externalized policy logic of the customer.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enforced by resource manager</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(such as WebSEAL or other blades)</td>
<td>Description</td>
</tr>
<tr>
<td>POP attribute</td>
<td>Description</td>
</tr>
<tr>
<td>Quality of protection</td>
<td>Specifies degree of data protection: none, integrity, privacy.</td>
</tr>
<tr>
<td>IP endpoint authentication method policy</td>
<td>Specifies authentication requirements for access from members of external networks.</td>
</tr>
</tbody>
</table>

The concept of inherited, or sparse ACLs as described in “Sparse security policy model” on page 54 also applies to POP policies in the same manner.

**Authorization rules**

An authorization rule policy specifies security policy that applies to an object based on a variety of conditions, such as context and environment. Each authorization rule policy has a unique name and can be applied to multiple objects within a domain.

Like ACLs and POPs, authorization rules are defined to specify conditions that must be met before access to a protected object is permitted. A rule is created using a number of Boolean conditions that are based on data supplied to the authorization engine within the user credential, from the resource manager, or from the encompassing business environment. The language of an authorization rule allows customers to work with complex, structured data, by examining the values in that data and making informed access decisions. This information can be defined statically within the system or defined during the course of a business process. Rules can also be used to implement extensible attribute-based authorization policy using attributes within the business environment or attributes from trusted external sources.

The rule is stored as a text rule within a rule policy object and is attached to a protected object in the same way and with similar constraints as ACLs and POPs.
Chapter 5. Domain management

An administrator in the management domain can create additional domains. A domain is given a unique name and a domain administrator must be specified when the domain is initially created. Domain administrators can perform administrative tasks only within their own domain and do not have authority to perform tasks in other domains.

Users, groups, and other objects also can be created within a domain. Users and groups are specific to their domain and are not allowed to access resources contained in other domains. If users and groups are created outside of Tivoli Access Manager these users and groups can be imported into another domain.

Resources are defined and access controls for resources protected by Tivoli Access Manager are maintained on a per domain basis and cannot be shared between domains.

Create a domain

Any number of additional domains can be created in addition to the management domain.

Only an administrator who is logged in to the management domain is authorized to create domains. A domain can be created only by an administrator with the appropriate privileges within the management domain.

To create a domain using the Web Portal Manager or the pdadmin command line utility, log in to the management domain as a domain administrator.

Web Portal Manager

To create a domain using Web Portal Manager:

1. Log in to the management domain.
2. Click Secure Domain + Create Secure Domain.
3. Type the Secure Domain Name that you want to create (for example, Domain-ABC). This field is required.
   Restrictions on the domain name:
   • Limited to 64 characters in length
   • Can contain a–z, A–Z, 0–9, hyphen ( - ), underscore ( _ ), period ( . ), at sign ( @ ), or ampersand ( & )
   • Can contain any character from a double-byte character set
4. Type a Description of the domain, such as: Test Domain.
5. Type a New Domain Administrator Id (for example, myadmin_id). This field is required.
   Note: You must create an administrator ID for the domain.
6. Type a New Administrator Password (for example, 12A345). Passwords must adhere to the password policies set by the domain administrator. This field is required.
7. Type the password again in Confirm Password. This field is required.
8. Click Create.
To log in to the domain you just created using Web Portal Manager:

1. From the login screen, type the domain name that you just created. The default domain name is Default.
2. Type the user ID that was created for this domain. The default user ID is sec_master.
3. Type the password associated with the user ID.

**pdadmin**

To create a domain using the pdadmin utility, log in to the management domain and use the `pdadmin domain create` command. For example, to create a domain named Domain-ABC, enter on one line, the following:

```
domain create Domain-ABC myadmin_id 12A345
-desc "Test Domain"
```

For example, for the myadmin_id administrator to log in interactively to the Domain-ABC domain using the `pdadmin login` command, enter:

```
padmin login -a myadmin_id -p 12A345 -d Domain-ABC
```

For more information, see the IBM Tivoli Access Manager for e-business Command Reference.

**List domains**

To list all domains, except for the management domain, using the Web Portal Manager or the pdadmin command line utility, log in to the management domain as a domain administrator.

Only an administrator who is logged in to the management domain is authorized to list domains. The administrator must have the appropriate privileges to list domains within the management domain.

**Web Portal Manager**

To list all domains, except for the management domain, using Web Portal Manager:

1. Log in to the management domain.
2. Click Secure Domain → List Secure Domain.

The Manage Secure Domains window displays all the domain names, except for the management domain, as links. Additionally, you can filter the domain names to view only the domain names that meet the criteria you specify.

**pdadmin**

To list all domains, except for the management domain, using the pdadmin utility, log in to the management domain and use the `pdadmin domain list` command.

```
padmin sec_master> domain list
```

For more information, see the IBM Tivoli Access Manager for e-business Command Reference.

**Delete a domain**

Only an administrator who is logged in to the management domain is authorized to delete domains. A domain can be deleted only by an administrator with the appropriate privileges within the management domain.
Deleting a domain deletes the specified Tivoli Access Manager group and optionally, if you use the `padmin group delete --registry` option, deletes the group’s information from the user registry. ACL entries associated with the group are also deleted.

**Note:** The delete operation cannot be reversed.

To delete a domain using the Web Portal Manager or the `padmin` command line utility, log in to the management domain as a domain administrator.

**Web Portal Manager**

To delete a domain using Web Portal Manager:

1. Log in to the management domain.
2. Click **Secure Domain → List Secure Domain**.
3. From the Domain List window select the domain you want to delete
4. From the Domain Properties window click **Delete**.

   To permanently remove domain information from the user registry, click **Delete Registry Entry**. Otherwise, the user and group information for the domain remains in the user registry and can be used again if the domain is recreated.

**padmin**

To delete a domain using the `padmin` utility, log in to the management domain and use the `padmin domain delete` command. To permanently remove domain information from the user registry, specify the `--registry` option. Otherwise, the user and group information for the domain remains in the user registry and can be used again if the domain is recreated.

For example, to delete the domain named Domain-ABC and permanently remove the domain information from the user registry, enter:

```
padmin sec_master> domain delete Domain-ABC --registry
```

**Note:** If you unconfigure the management domain using the `pdconfig` utility, instead of using the `padmin domain delete` command, any additional domains that exist also are deleted.

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*.

---

**Modify a domain**

Only an administrator who is logged in to the management domain is authorized to modify a domain description. A domain can be modified only by an administrator with the appropriate privileges within the management domain.

To modify a domain description using the Web Portal Manager or the `padmin` command line utility, log in to the management domain as a domain administrator.

**Web Portal Manager**

To modify a domain description using Web Portal Manager:

1. Log in to the management domain.
2. Click **Secure Domain → List Secure Domain**.
3. From the Manage Secure Domains window, click the name of the domain that you want to change (for example, Domain-ABC.

4. From the Secure Domain Properties window, edit the Description field to add a new description or change the existing description. For example, enter new test domain description to change the existing description.

5. Click Apply.

**pdadmin**

To modify a domain description using the pdadmin utility, log in to the management domain and enter, on one line, the `pdadmin domain modify` command.

```
padmin sec_master> domain modify Domain-ABC description "new test domain description"
```
Chapter 6. Object space management

Tivoli Access Manager represents resources to be protected using a virtual representation called the object space or the protected object space. An object space consists of resource objects and container objects. Resource objects are the logical representation of resources to be protected. Container objects allow you to group resource objects and other container objects hierarchically into logical groups or regions. Grouping similar objects together makes it easier for you to administer a consistent security policy.

Security policy is applied by attaching access control lists (ACLs), protected object policies (POPs), and authorization rules to the objects within the object space that represent the physical resources to be protected. The Tivoli Access Manager authorization service makes decisions to permit or deny access to resources based on user credentials and the conditions specified by the security policy.

The Tivoli Access Manager security model depends on ACLs, POPs, and authorization rules to provide fine-grained protection for these resources. A corporate security policy is implemented by the strategically applying custom ACL and POP policies to those resources requiring protection. The Tivoli Access Manager authorizations service makes decisions to permit or deny access to resources based on user credentials and the specific permissions and conditions set in the ACL and POP policies.

In order to apply ACLs, POPs, and authorization rules and allow the authorizations service to perform its security checks, Tivoli Access Manager uses a virtual object representation of domain resources called the protected object space.

As a domain administrator, you can use the Web Portal Manager or the pdadmin command line utility to attach an ACL, POP, or authorization rule to the objects in the object space.

Create an object space

To create an object space using the Web Portal Manager or the pdadmin command line utility, log in to the desired domain as the domain administrator.

Web Portal Manager

To create an object space using Web Portal Manager:
1. Log in to the domain.
2. Click Object Space + Create Object Space.
3. Type an Object Name. This field is required. For example: /Test-Space
4. Type a Description for the object space. For example: New Object Space
5. Click Create. To see the /Test-Space object space in the hierarchical structure, browse the object space. See "Browse the object space" on page 67.

Because an object space consists of resource objects and container objects, you do not have to specify any object type when using Web Portal Manager.
pdadmin

To create an object space in the domain using the pdadmin utility, log in to the domain and use the pdadmin objectspace create command.

Note: Do not use the pdadmin objectspace command on object spaces created by or developed by using Tivoli Access Manager. Tivoli Access Manager object spaces include /Management, /WebSEAL, /OSSEAL, and /PDMQ.

For example, to create the object space named /Test-Space that is an application container object, enter the following:

```
pdadmin sec_master>
objectspace create /Test-Space "New Object Space" 14
```

When creating an object space, an object type must be specified. This object space example assigns a category number of 14, which is for an application container object.

"Protected object space“ on page 34 discusses the two general types of objects: resource objects and container objects. You can select any of the listed object space types, or use any unused category number listed in the following table to designate the object space type and assign a meaning to it.

These object space types are valid for Tivoli Access Manager:

<table>
<thead>
<tr>
<th>Object Types</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – unknown</td>
<td>9 – HTTP server</td>
</tr>
<tr>
<td>1 – secure domain</td>
<td>10 – nonexistent object</td>
</tr>
<tr>
<td>2 – file</td>
<td>11 – container object</td>
</tr>
<tr>
<td>3 – executable program</td>
<td>12 – leaf object</td>
</tr>
<tr>
<td>4 – directory</td>
<td>13 – port</td>
</tr>
<tr>
<td>5 – junction</td>
<td>14 – application container object</td>
</tr>
<tr>
<td>6 – WebSEAL server</td>
<td>15 – application leaf object</td>
</tr>
<tr>
<td>7 – unused</td>
<td>16 – management object</td>
</tr>
<tr>
<td>8 – unused</td>
<td>17 – unused</td>
</tr>
</tbody>
</table>

For more information, see the IBM Tivoli Access Manager for e-business Command Reference.

Create an object

To create an object using the Web Portal Manager or the pdadmin command line utility, log in to the desired domain as the domain administrator.

Web Portal Manager

To create an object using Web Portal Manager:

1. Log in to the domain.
2. Click Object Space → Create Object.
3. Type an Object Name. This field is required. Type the full path for the object name. For example: /Management/Groups/Travel
4. Type a Description for the object space. For example: Travel Container Object
5. Click Create.
6. To see the new object in the hierarchical structure, browse the object space. Click Object Space → Browse Object Space → expand, as necessary.
7. To select the Can Policy be attached to this object check box on the Protected Object Properties window, first select an object.

Because a Tivoli Access Manager object already has resource objects and container objects defined, you do not have to specify any object type when using Web Portal Manager. Different icons are used to represent the different types of object.

**pdadmin**

To create an object in the object space using the `pdadmin` utility, log in to the domain and use the `pdadmin object create` command.

For example, to create the object named /Travel that is an application container object, enter on one line the following command:

```
pdadmin sec_master> object create /Management/Groups/Travel "Travel Container Object" 14
   ispolicyattachable yes
```

When creating an object using the `pdadmin` utility, an object type must be specified. This object example assigns a category number of 14, which is for an application container object.

"Protected object space" on page 34 discusses the two general types of objects: resource objects and container objects. You can select any of the listed object types, or use any unused category number listed in the following table to designate the object type and assign a meaning to it.

|-----------------------|-----------------|---------------------|-------------|--------------------------|----------------|--------------|---------------------|-------------|-------------|---------------------|----------------------------|------------------------|----------------------|-----------|-------------------------------------|--------------------------------|----------------------|-----------|

The `ispolicyattachable {yes|no}` parameter determines whether you can attach an ACL policy to this object.

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*.

**Browse the object space**

To list all object spaces using the Web Portal Manager or the `pdadmin` command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**

To list all object spaces using Web Portal Manager:

1. Log in to the domain.
2. Click Object Space → Browse Object Space.

The Browse Object Space window displays all object spaces in the domain in a hierarchical structure. All object spaces appear at the same hierarchical structure.
level as the default /Management object space. Each object space displays as a link. When selected, the link displays the Protected Object Properties window.

**pdadmin**

To list all object spaces in the domain using the **pdadmin** utility, log in to the domain and use the **pdadmin objectspace list** command.

```bash
pdadmin sec_master>
objectspace list
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*.

**Delete an object space**

To delete an object from the object space using the Web Portal Manager or the **pdadmin** command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**

To delete an object from the object space using Web Portal Manager:

1. Log in to the domain.
2. Click **Object Space > Browse Object Space**.
3. From the Browse Object Space window, expand and click the protected object that you want to delete.
4. From the Protected Object Properties window, the name of the object space is displayed in the **Object Name** field. Click **Delete**.
5. To confirm the deletion, click **Delete** again. If successful, a message displays indicating that the object has been deleted.

**pdadmin**

To delete an object space in the domain using the **pdadmin** utility, log in to the domain and use the **pdadmin objectspace delete** command. For example, to delete the object space named /Test-Space, enter the following:

```bash
pdadmin sec_master>
objectspace delete /Test-Space
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*.
Chapter 7. Protected object management

A protected object is a logical representation of an actual system resource that is used for applying access control lists (ACLs), protected object policies (POPs), and authorization rules for authorizing user access.

As a domain administrator, you can use the Web Portal Manager or the pdadmin command line utility to create and delete objects.

As soon as an object space has been created, you can populate it with objects. For information about creating an object space, see “Create an object space” on page 65.

Create an object

To create an object using the Web Portal Manager or the pdadmin command line utility, log in to the desired domain as a domain administrator.

Web Portal Manager

To create a new object using Web Portal Manager:
1. Log in to the domain.
2. Click Object Space → Create Object.
3. Type the full path of the object in the Object Name field (for example, /Management/Groups/test-object). This field is required.
4. Optionally, type a description for the object (for example, Test Object).
5. Click Create.
6. If you want to be able to attach a policy to this protected object, click Object Space → Browse Object Space. The Browse Object Space window provides a hierarchical display of all objects in the domain as links. Click on the object to go to the Protected Object Properties window, and then select the Can Policy be attached to this object check box.

pdadmin

To create an object in the domain using the pdadmin utility, log in to the domain and use the pdadmin object create command. For example, to create the object named /Management/test-object that is an application container object, enter the following:

    pdadmin object create /Management/test-object "Test Object" 14 ispolicyattachable yes
The type can be one of the following categories:

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – unknown</td>
<td></td>
</tr>
<tr>
<td>1 – secure domain</td>
<td></td>
</tr>
<tr>
<td>2 – file</td>
<td></td>
</tr>
<tr>
<td>3 – executable program</td>
<td></td>
</tr>
<tr>
<td>4 – directory</td>
<td></td>
</tr>
<tr>
<td>5 – junction</td>
<td></td>
</tr>
<tr>
<td>6 – WebSEAL server</td>
<td></td>
</tr>
<tr>
<td>7 – unused</td>
<td></td>
</tr>
<tr>
<td>8 – unused</td>
<td></td>
</tr>
<tr>
<td>9 – HTTP server</td>
<td></td>
</tr>
<tr>
<td>10 – nonexistent object</td>
<td></td>
</tr>
<tr>
<td>11 – container object</td>
<td></td>
</tr>
<tr>
<td>12 – leaf object</td>
<td></td>
</tr>
<tr>
<td>13 – port</td>
<td></td>
</tr>
<tr>
<td>14 – application container object</td>
<td></td>
</tr>
<tr>
<td>15 – application leaf object</td>
<td></td>
</tr>
<tr>
<td>16 – management object</td>
<td></td>
</tr>
<tr>
<td>17 – unused</td>
<td></td>
</tr>
</tbody>
</table>

When creating an object, a type must be specified. You can select an appropriate category, or use any number to designate the object type and assign a meaning to it.

If the `ispolicyattachable` field is omitted from the `pdadmin object create` command, the utility assumes that you intended to use the `objectspace create` command. An object space is created rather than an object.

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*.

**List objects**

To list objects in the domain using the Web Portal Manager or the `pdadmin` command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**

To list objects using Web Portal Manager:

1. Log in to the domain.
2. Click **Object Space → Browse Object Space**.

   The Browse Object Space window provides a hierarchical display of all objects in the domain as links.

**pdadmin**

To list all objects in the domain using the `pdadmin` utility, log in to the domain and use the `pdadmin object list` command.

For example, to list the object under the `/Management` object space, enter:

```
pdadmin sec_master> object list /Management
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*.

**Delete an object**

To delete an object using the Web Portal Manager or the `pdadmin` command line utility, log in to the desired domain as a domain administrator.
Web Portal Manager

To delete an object using Web Portal Manager:

1. Log in to the domain.
2. Click **Object Space + Browse Object Space**.
   The Browse Object Space window provides a hierarchical display of all objects in the domain as links.
3. Click the link for an object (for example, `/Management/test-object`) to see its properties, such as whether ACLs, POPs, and authorization rules are attached to the object and whether the object has any extended attributes.
4. From the Protected Object Properties window, ensure the object named is the one you want to deleted, then click **Delete**.

pdadmin

To delete an object in the domain using the **pdadmin** utility, log in to the domain and use the **pdadmin object delete** command. For example, to delete the object named `/Management/test-object`, enter the following:

`pdadmin object delete /Management/test-object`

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*. 
Chapter 8. Access control list management

You can use access control list (ACL) policies to control access to domain resources. Using the pdadmin command line utility or the Web Portal Manager, a domain administrator can create ACL policies that consist of ACL entries to protect the resources defined in the administrator’s domain. Also, the domain administrator can choose for another user to have administrative authority by setting the ACL entries for the user to match the domain administrator’s ACL entries. Now the user will have the same authority as the administrator.

ACL policies

An ACL policy consists of one or more ACL entries that describe:
- The names of users and groups whose access to the object is explicitly controlled
- The specific operations permitted to each user, group, or role
- The specific operations permitted to the special any-other and unauthenticated user categories

![ACL policy example](image)

Figure 17. Access control list for a Web page object

You can use the pdadmin command line utility or the Web Portal Manager to create, modify, and delete ACL entries.

ACL entries

An ACL entry contains either two or three attributes, depending on the ACL entry type, and appears in the following format:

![ACL entry example](image)

Figure 18. ACL entry attributes

- **Type** – the entity category (user or group) for which the ACL was created
- **ID (Identity)** – the unique identifier (name) of the entity
The **any-other** and **unauthenticated** ACL entry types do not require the ID attribute.

- **Permissions (or actions)** – the set of operations that are permitted on the object by this user or group.

Most permissions dictate the client’s ability to perform a specific operation on the resource.

In the above example, the user **adam** (type = user, ID = adam) has permission to read (view) the object protected by this ACL policy. The read (r) permission allows the read operation. The traverse (T) permission enforces the traverse rule.

**Type attribute**

An ACL entry type identifies the user, group, or special entity for a specific ACL entry. There are four ACL entry types.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>user</strong></td>
<td>Sets permissions for a specific user in a domain. The user must be a member of the domain with an account in the registry. The user entry type requires a user name (ID). The entry format is: user ID permissions</td>
</tr>
<tr>
<td></td>
<td>For example: user anthony -------T-----r-</td>
</tr>
<tr>
<td><strong>group</strong></td>
<td>Sets permissions for all members of a specific group in a domain. The group entry type requires a group name (ID). The entry format is: group ID permissions</td>
</tr>
<tr>
<td></td>
<td>For example: group engineering -------T-----r-</td>
</tr>
<tr>
<td><strong>any-other</strong></td>
<td>Sets permissions for all authenticated users. No ID designation is required. The entry format is: any-other permissions</td>
</tr>
<tr>
<td>(also known as any-authenticated)</td>
<td>For example: any-other -------T-----r-</td>
</tr>
<tr>
<td><strong>unauthenticated</strong></td>
<td>Sets permissions for those users who have not been authenticated by the policy server. No ID designation is required. The entry format is: unauthenticated permissions</td>
</tr>
<tr>
<td></td>
<td>For example: unauthenticated -------T-----r-</td>
</tr>
</tbody>
</table>

This ACL entry is a mask (a bit-wise “and” operation) against the **any-other** ACL entry to determine the permission set. A permission for **unauthenticated** is granted only if the permission also appears in the **any-other** entry. For example, the following **unauthenticated** ACL entry:

unauthenticated ------------------rw

masked against this **any-other** ACL entry:

any-other -------T-----r-

results in these permissions:

--------------r- (read only).
ID attribute

The ACL entry ID is the unique identifier, or name, for a user or group entry type. IDs must represent valid users, groups, or both that are created in a Tivoli Access Manager domain.

Examples:
user michael
user anthony
group engineering
group documentation
group accounting

Note: The any-other and unauthenticated ACL entry types do not use the ID attribute.

Permissions (actions) attribute

Each ACL entry contains a set of permissions (or actions) that describes the specific operations that are permitted on the object by the user or group.

ACL policies control protected resources in the following ways:
• A user’s ability to perform operations on protected objects
• An administrator’s ability to change access control rules on the object and any sub-objects
• Tivoli Access Manager’s ability to delegate user’s credentials

Note: ACL permissions are context-sensitive — the behavior of certain permissions varies according to the region of the protected object space in which they are applied. For example, the modify (m) permission has a different meaning for protected resources within the /WebSEAL object space than for protected resources within the /Management object space.

Default Tivoli Access Manager permissions (actions)

Tivoli Access Manager defines seventeen default permissions (actions). The Web Portal Manager divides these permissions into three categories.

<table>
<thead>
<tr>
<th>Action Bit</th>
<th>Description</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Attach</td>
<td>Base</td>
</tr>
<tr>
<td>A</td>
<td>Add</td>
<td>Base</td>
</tr>
<tr>
<td>b</td>
<td>Browse</td>
<td>Base</td>
</tr>
<tr>
<td>B</td>
<td>Bypass protected object policy (POP)</td>
<td>Base</td>
</tr>
<tr>
<td>c</td>
<td>Control</td>
<td>Base</td>
</tr>
<tr>
<td>d</td>
<td>Delete</td>
<td>Generic</td>
</tr>
<tr>
<td>g</td>
<td>Delegation</td>
<td>Base</td>
</tr>
<tr>
<td>l</td>
<td>List directory</td>
<td>Application</td>
</tr>
<tr>
<td>m</td>
<td>Modify</td>
<td>Generic</td>
</tr>
<tr>
<td>N</td>
<td>Create</td>
<td>Base</td>
</tr>
<tr>
<td>r</td>
<td>Read</td>
<td>Application</td>
</tr>
</tbody>
</table>
### Tivoli Access Manager provides the capability to define many more additional actions, or permissions, for use by applications. For more information, see "Action group management" on page 81.

### Managing an access control list

You create and configure an ACL and then attach the ACL to objects in the protected object space. ACL policies are placed in the master authorization database on a per-domain basis, which is controlled by the policy server.

You can perform the following ACL tasks:

- **“Create an ACL”**
- **“List ACLs”** on page 77
- **“Delete an ACL”** on page 77
- **“Modify an ACL”** on page 77
- **“Applying an ACL attribute to a protected object”** on page 78
  - “Attach an ACL to an object” on page 78
  - “Find where an ACL is attached” on page 79
  - “Detaching an ACL” on page 79

Protected object policies (POPs) operate in a similar way to ACL policies. POP policies are inherited in the same way as ACL policies. See "Managing protected object policies" on page 88 for more information about POPs.

### Create an ACL

To create an ACL by using the Web Portal Manager or the `pdadmin` command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**

To create an ACL using Web Portal Manager:

1. Log in to the domain.
2. Click **ACL › Create ACL**.
3. Type an **ACL Name** (for example, Test-ACL). This field is required.
4. Optionally, type a **Description** of the ACL (for example, Test of new ACL).
5. Click **Create**. If successful, a link for this ACL will be available when you list all the ACLs.

<table>
<thead>
<tr>
<th>Action Bit</th>
<th>Description</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>Server administration</td>
<td>Generic</td>
</tr>
<tr>
<td>t</td>
<td>Trace</td>
<td>Base</td>
</tr>
<tr>
<td>T</td>
<td>Traverse</td>
<td>Base</td>
</tr>
<tr>
<td>v</td>
<td>View</td>
<td>Generic</td>
</tr>
<tr>
<td>W</td>
<td>Password</td>
<td>Base</td>
</tr>
<tr>
<td>x</td>
<td>Execute</td>
<td>Application</td>
</tr>
<tr>
<td>R</td>
<td>Bypass rule</td>
<td>Base</td>
</tr>
</tbody>
</table>
**pdadmin**
To create a new ACL in the domain by using the **pdadmin** utility, log in to the domain and use the **pdadmin acl create** command. For example, to create an ACL named Test-ACL, enter the following:

```
pdadmin sec_master> acl create Test-ACL
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference.*

**List ACLs**
To list all ACLs by using the Web Portal Manager or the **pdadmin** command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**
To list all ACLs in the domain using Web Portal Manager:

1. Log in to the domain.
2. Click ACL → List ACL.
   
The Manage ACLs window displays all the ACLs in the domains.

**pdadmin**
To list all ACLs in the domain by using the **pdadmin** utility, log in to the domain and use the **pdadmin acl list** command.

```
pdadmin sec_master> acl list
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference.*

**Delete an ACL**
To delete an ACL using the Web Portal Manager or the **pdadmin** command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**
To delete an ACL using Web Portal Manager:

1. Log in to the domain.
2. Click ACL → List ACL.
3. From the Manage ACLs window, select one or more check boxes of the ACLs that you want to delete.
4. Click Delete, and then confirm the deletion by clicking Delete on the Delete ACLs confirmation window. If successful, the ACLs will no longer included in the list of ACLs in the Manage ACLs window.

**pdadmin**
To delete an ACL in the domain using the **pdadmin** utility, log in to the domain and use the **pdadmin acl delete** command. For example, to delete the ACL named Test-ACL, enter the following:

```
pdadmin sec_master> acl delete Test-ACL
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference.*

**Modify an ACL**
To modify an ACL by using the Web Portal Manager or the **pdadmin** command line utility, log in to the desired domain as a domain administrator.
Web Portal Manager
To modify an ACL using Web Portal Manager:
1. Log in to the domain.
2. Click ACL → List ACL.
3. Click the link for the ACL you want to change (for example, Test-ACL).
4. From the ACL Properties window, click the General tab to view or change the description of the ACL, view a list of the ACL entries, and create or delete ACL entries. For example, to change the description from Test ACL, type Test 1 for ACL and then click Set next to the Description field.
   For online help, click the question mark to open a separate help window for the current window.
5. Click on a permissions link to view a list of, enable, or disable permissions that are associated with the selected ACL.
   The ACL Entry Properties window is displayed. From this window, you can select or clear the check boxes for the permissions for the ACL, and then click Apply.
6. Click the Attach tab to view a list of, attach the ACL to, or detach the ACL from a protected object.
7. Click the Extended Attributes tab to view a list of, create or delete extended attributes for the specified ACL.

pdadmin
To modify an ACL entry in the domain using the pdadmin utility, log in to the domain and use the pdadmin acl modify command. For example, to modify the ACL named Test-ACL for user named maryj to have r (read) permissions, enter the following:
```
    pdadmin sec_master> acl modify Test-ACL set user maryj r
```

To show the modifications to the ACL, use the pdadmin acl show command:
```
    pdadmin sec_master> acl show Test-ACL
    ACL Name: Test-ACL
    Description: 
    Entries: 
        User maryj r
```

For more information, see the IBM Tivoli Access Manager for e-business Command Reference.

Applying an ACL attribute to a protected object
You can apply ACL policies by attaching an ACL to a protected object. ACLs can also be detached from a protected object.

Attach an ACL to an object
To attach an ACL to a protected object using the Web Portal Manager or the pdadmin command line utility, log in to the desired domain as a domain administrator.

Web Portal Manager
To attach an ACL to a protected object using Web Portal Manager:
1. Log in to the domain.
2. Click ACL → List ACL.
3. Click the link for the name of the ACL that you want to change (for example, Test-ACL).
4. From the ACL Properties window, click the Attach tab.
5. Click Attach to display the Attach ACL window.
6. Type a Protected Object Path (for example, /Management/test-object). The object must already exist.
7. Click Attach. If successful, the protected object is displayed as a protected object link for the named ACL.

**pdadmin**
To attach an ACL to a protected object in the domain using the **pdadmin** utility, log in to the domain, and use the **pdadmin acl attach** command. For example, to attach an ACL named Test-ACL to a protected object named /Management/test-object, enter the following:

```
padmin sec_master> acl attach /Management/test-object Test-ACL
```

For more information, see the IBM Tivoli Access Manager for e-business Command Reference.

**Find where an ACL is attached**
To find where an ACL is attached using the Web Portal Manager or the **pdadmin** command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**
To find where an ACL is attached using Web Portal Manager:
1. Log in to the domain.
2. Click ACL → List ACL. A list of ACL names is displayed. Each ACL name is a link that you can click to display the ACL properties window.
3. Click the Attach tab.

**pdadmin**
To find where an ACL is attached in the domain using the **pdadmin** utility, log in to the domain and use the **pdadmin acl find** command. For example, to find where the ACL named Test-ACL is attached, enter the following:
```
padmin sec_master> acl find Test-ACL
```

For more information, see the IBM Tivoli Access Manager for e-business Command Reference.

**Detaching an ACL**
To detach an ACL from an object using the Web Portal Manager or the **pdadmin** command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**
To detach an ACL from a protected object in the domain using Web Portal Manager:
1. Log in to the domain.
2. Click ACL → List ACL.
3. From the Modify ACLs window, click the link for the ACL (for example, Test-ACL).
4. From the ACL Properties window, click the Attach tab.
5. If ACLs are currently attached to protected objects, select one or more check boxes for the ACLs objects that you want to detach.
6. Click **Detach**. You will be asked to confirm the detachment.

**pdadmin**
To detach an ACL from a protected object in the domain by using the **pdadmin** utility, log in to the domain, and use the **pdadmin acl detach** command. For example, to detach the ACL from the protected object named /Management/test-object, enter the following:

```
pdadmin sec_master> acl detach /Management/test-object
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*.

### Example ACL entries

You set permissions for specific users and groups by specifying the appropriate ACL entry type. In the following example, the group **documentation** has full access privileges:

```
group documentation --bcg--Tdmsv--lrx
```

You can restrict access to other authenticated users in the domain (not belonging to the documentation group) by using the **any-other** entry type:

```
any-other ---------T---------rx
```

You can further restrict access to the **unauthenticated** entry type for users who are not members of the domain.

```
unauthenticated ---------T----------r-
```

**Note:** Without an **unauthenticated** ACL entry, unauthenticated users cannot access any secure documents within the domain.

### ACL policies and the protected object space

Container objects represent specific regions of the protected object space and serve two important security functions:

- You can use the container object’s ACL to define high level policy for all subobjects within the region when no other explicit ACLs are applied.
- You can quickly deny access to all objects in a region by removing the traverse permission from the container object’s ACL.

#### Root (/) container object

The following security considerations apply for the **root** object:

- The **root** object begins the chain of ACL inheritance for the entire protected object space.
- The **root** object defines (through inheritance) the security policy for the entire object space If you do not apply any other explicit ACLs.
- The traverse (T) permission is required for access to any object below **root**.

#### The traverse permission

The traverse permission is a generic permission that applies throughout the protected object space.
### Object and object space permissions

The commands listed in the following table allow administration users to manage new objects and object spaces. Action tasks and associated permissions include:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>attach</td>
</tr>
<tr>
<td></td>
<td>Attach ACL or POP policies to objects; remove ACL or POP policies from objects.</td>
</tr>
<tr>
<td></td>
<td>pdadmin acl attach</td>
</tr>
<tr>
<td></td>
<td>pdadmin acl detach</td>
</tr>
<tr>
<td></td>
<td>pdadmin pop attach</td>
</tr>
<tr>
<td></td>
<td>pdadmin pop detach</td>
</tr>
<tr>
<td>b</td>
<td>browse</td>
</tr>
<tr>
<td></td>
<td>object space list</td>
</tr>
<tr>
<td></td>
<td>object list</td>
</tr>
<tr>
<td></td>
<td>object list and show</td>
</tr>
<tr>
<td></td>
<td>(additionally requires v)</td>
</tr>
<tr>
<td>d</td>
<td>delete</td>
</tr>
<tr>
<td></td>
<td>object space delete</td>
</tr>
<tr>
<td></td>
<td>object delete</td>
</tr>
<tr>
<td></td>
<td>object modify set name</td>
</tr>
<tr>
<td></td>
<td>(additionally requires m)</td>
</tr>
<tr>
<td>m</td>
<td>modify</td>
</tr>
<tr>
<td></td>
<td>object space create</td>
</tr>
<tr>
<td></td>
<td>object create</td>
</tr>
<tr>
<td></td>
<td>object modify</td>
</tr>
<tr>
<td>v</td>
<td>view</td>
</tr>
<tr>
<td></td>
<td>object show</td>
</tr>
<tr>
<td></td>
<td>object list and show</td>
</tr>
<tr>
<td></td>
<td>(additionally requires b)</td>
</tr>
</tbody>
</table>

**Note:** The following operations require both browse (b) and view (v) permissions:

- `pdadmin object list`
- `pdadmin object list and show`

### Action group management

Actions are used to grant permission to perform a specific operation on resources protected by Tivoli Access Manager. Seventeen actions are predefined for immediate use, as described in “Default Tivoli Access Manager permissions (actions)” on page 75. Tivoli Access Manager provides the ability to create resource manager-specific actions. For example, Tivoli Access Manager for Business Integration defines Enqueue and Dequeue actions to grant permission to put messages onto a message queue or to get messages off of a queue.
**Action groups**

This section describes how to create action groups that serve as containers for an expanded set of custom actions:

- Each action group is capable of holding up to 32 action bits.
- An action bit is made up of a letter: a-z, A-Z.
- Each action bit character can be used only once within an action group.
- You can reuse the same action bit in other action groups.
- The default Tivoli Access Manager actions are stored in an initial predefined action group called *primary*, as shown in Figure 19.

![Primary action group diagram](image1)

*Figure 19. Primary action group*

Tivoli Access Manager supports a total of 32 action groups (including the primary action group) for a total of 1024 individual actions.

![Multiple action groups diagram](image2)

*Figure 20. Multiple action groups*

**Create a new action group**

To create an action group using the Web Portal Manager or the pdadmin command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**

To create an action group using Web Portal Manager:

1. Log in to the domain.
2. Click ACL → Create Action Group.
3. Type the new **Action Group Name** (for example, test-group. This field is required.
   
   Click **Create**. If successful, a message is displayed when the action group is created.
**pdadmin**
To create an action group in the domain using the `pdadmin` utility, log in to the domain and use the `pdadmin action group create` command. For example, to create a new action group named `test-group`, enter the following:
```
pdadmin sec_master> action group create test-group
```
The default primary action group always appears in a group listing and cannot be deleted.

You must have an entry in an ACL on the `/Management/ACL` object with the modify (m) permission to create action groups and the delete (d) permission to delete action groups.

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*.

**List action groups**
To list all action groups using the Web Portal Manager or the `pdadmin` command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**
To list all action group using Web Portal Manager:
1. Log in to the domain.
2. Click ACL → List Action Groups.
   The Manage Action Groups window displays a list of all action groups in the domain.

**pdadmin**
To list all action groups in the domain using the `pdadmin` utility, log in to the domain and use the `pdadmin action group list` command.
```
pdadmin sec_master> action group list
```
For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*.

**Delete an action group**
To delete an action group using the Web Portal Manager or the `pdadmin` command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**
To delete an action group using Web Portal Manager:
1. Log in to the domain.
2. Click ACL → List Action Groups.
3. From the Manage Action Groups window, select one or more check boxes for the action groups that you want to delete.
4. Click Delete.
5. Confirm the deletion by clicking Delete on the Delete Action Groups window.

**pdadmin**
To list action groups in the domain using the `pdadmin` utility, log in to the domain and use the `pdadmin action group delete` command.
```
pdadmin sec_master> action group delete test-group
```
For more information, see the IBM Tivoli Access Manager for e-business Command Reference.

Create new actions in an action group
To create an action in an action group using the Web Portal Manager or the pdadmin command line utility, log in to the desired domain as a domain administrator.

Web Portal Manager
To create an action within an action group using Web Portal Manager:
1. Log in to the domain.
2. Click ACL > List Action Groups.
3. Click on the link for the action group name (for example, Test-Group) that you want to create the action in. The Action Group Properties window is displayed, and the Action Group Name is automatically filled in.
4. From the Action Group Properties window, click Create to display the Create Action window. The Action Group Name is automatically filled in.
5. Type a single character Action Name (for example, x). This field is required.
6. In the Action Label field, type a short description of the action (for example, Execute). This field is required.
7. In the Action Type field, type a description of the action, such as to which application the action is specific (for example, WebSEAL). This field is required.
8. Click Create. If successful, a message is displayed when the action is created.

pdadmin
To create an action within an action group in the domain using the pdadmin utility, log in to the domain, and use the pdadmin action create command. To create an x action within an action group named Test-Group, enter the following:

```
sec_master> action create x Execute WebSEAL Test-Group
```

For more information, see the IBM Tivoli Access Manager for e-business Command Reference.

Custom actions
Default Tivoli Access Manager actions are available to all applications. If an application makes use of a default Tivoli Access Manager action, the associated operation should very closely match that of the actual operation normally performed by Tivoli Access Manager. For example, r should be used only by an operation that requires a read-only access to a protected object.

Note: An application can use a default Tivoli Access Manager action for a completely unrelated operation because the authorization service does not know or care about the operation. However, this situation would cause difficulty for an administrator who would have to distinguish between two dissimilar uses of the same action.

If an application uses an operation that is not well represented by any of the default actions, Tivoli Access Manager allows you to define a new action that can be used by this application and be recognized by the authorization service.

See “Action group management” on page 81.
**Custom action example**

In this example, as shown in [Figure 21](#), there is a requirement to protect a certain printer device from unauthorized use. A print spooling service is written with the authorization application programming interface (authorization API) so that it can call the authorization service to perform ACL checks on requests made to the printer.

The standard Tivoli Access Manager actions do not include an obvious permission for protecting printers. However, the printer can be protected by a newly created action (p in this example).

An ACL policy is attached to the printer object. If a user requests the use of the protected printer, that user must have an ACL entry containing the p permission. The authorization service returns a favorable response if the p permission is present and the printing operation proceeds. If the authorization service finds no existence of a p permission for that user, the printing operation is not allowed to proceed.

![Figure 21. Custom print spooler action](image)

**Entering custom actions into ACL entries**

As discussed in “ACL entries” on page 73, ACL entries contain an entry type, a type ID (for user and group types), and the set of permitted action bits.

You must use a special syntax to identify custom action bits belonging to action groups other than the “primary” action group. Action strings that represent the action bits from multiple action groups are presented in the following format:

`action...action[action-group]action...action,,,`

For example:

`abgTr[groupA]Pq[groupB]Rsy[groupC]ab`

- The first set of action bits (abgTr) represent actions from the “primary” (Tivoli Access Manager default) action group.
- Action group A contains actions P and q.
- Action group B contains actions R, s, and y.
- Action group C contains actions a and b.
- Note that action group C contains action bits that use the same letters as action bits in the “primary” group.

Because the action bits are associated with a specific action group (C), the a and b action bits have unique identities and can represent very different actions from the a and b action bits in the “primary” action group.
Example

Show action groups

```
pdadmin sec_master> action group list

primary
test-group
```

List actions in action group “test-group”

```
pdadmin sec_master> action list test-group

P Test-Action Special
S Test-Action2 Special
```

List ACL policies

```
pdadmin sec_master> acl list

default-webseal
default-root
default-gso
default-policy
default-config
test
default-replica
default-management
```

Show details of ACL “test”

```
pdadmin sec_master> acl show test

ACL Name: test
Description:
Entries:
    User sec_master Tcmdbva
    Group ivmgrd-servers Tl
    Any-other r
```

Add ACL entry for user Kathy containing actions from action groups “primary” and “test-group”

```
pdadmin sec_master> acl modify test set user kathy brT[test-group]PS
```

```
pdadmin sec_master> acl show test

ACL Name: test
Description:
Entries:
    User sec_master Tcmdbva
    Group ivmgrd-servers Tl
    Any-other r
    User kathy Tbr[test-group]PS
```
Chapter 9. Protected object policy management

While the access control list (ACL) policies provide the authorization service with information to make a yes or no answer on a request to access a protected object and perform some operation on that object, a protected object policy (POP) contains additional conditions on the request that are passed back to the resource manager along with the yes ACL policy decision from the authorizations service. It is the responsibility of Tivoli Access Manager and the resource manager to enforce the POP conditions.

Table 1 lists the available attributes for a Tivoli Access Manager POP where conditions are enforced by Tivoli Access Manager:

<table>
<thead>
<tr>
<th>POP attribute</th>
<th>Description</th>
<th>pdadmin pop commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the policy. This becomes the pop-name in the pdadmin pop online commands.</td>
<td>create delete</td>
</tr>
<tr>
<td>Description</td>
<td>Descriptive text for the policy. This appears in the pop show command.</td>
<td>modify set description</td>
</tr>
<tr>
<td>Warning mode</td>
<td>Provides administrators a means to test security policy.</td>
<td>modify set warning</td>
</tr>
<tr>
<td>Audit level</td>
<td>Specifies type of auditing: all, none, successful access, denied access, errors.</td>
<td>modify set audit-level</td>
</tr>
<tr>
<td>Time-of-day access</td>
<td>Day and time restrictions for successful access to the protected object.</td>
<td>modify set tod-access</td>
</tr>
<tr>
<td>Extended attributes</td>
<td>Specifies supplemental data fields.</td>
<td>modify set attribute modify delete attribute list attribute show attribute</td>
</tr>
</tbody>
</table>

Table 2 lists the available attributes for a Tivoli Access Manager POP where conditions are enforced by the resource manager (for example, WebSEAL):

<table>
<thead>
<tr>
<th>POP attribute</th>
<th>Description</th>
<th>pdadmin pop commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality of protection</td>
<td>Specifies degree of data protection: none, integrity, privacy.</td>
<td>modify set qop</td>
</tr>
<tr>
<td>IP endpoint authentication method policy</td>
<td>Specifies authentication requirements for access from members of external networks.</td>
<td>modify set ipauth add modify set ipauth remove modify set ipauth anyotherw</td>
</tr>
</tbody>
</table>
Notes:
1. The Tivoli Access Manager Base component provides IP endpoint authorization policy; WebSEAL provides IP endpoint authentication method policy.
2. For Tivoli Access Manager IP address support:
   • You can grant access to a protected resource based on the IP address that is used by the identity. For example, only users from IP address 9.18.n.n are allowed to access the protected resource.
   • You can define that an additional authentication level is required to access this protected resource based on the IP address that is used by the identity. The step-up level authentication is described in “Configuring levels for step-up authentication” on page 97 and the IBM Tivoli Access Manager for e-business WebSEAL Administration Guide.
3. The time-of-day access and the IP endpoint authentication method access place restrictions on the access to the object.
4. Audit level and quality of protection inform the authorizations service that extra services are required when permitting access to the object.
5. Warning mode provides a way to test security policy before they are made active.

Managing protected object policies

You create and configure a protected object policies (POP) and then attach the POP to objects in the protected object space. POP policies are placed in the master authorization database on a per domain basis, which is controlled by the policy server.

The following POP tasks can be performed:
• “Create a POP”
• “List POPs” on page 90
• “Delete a POP” on page 90
• “Modify a POP” on page 91

The following tasks to apply POP attributes to protected objects can be performed:
• “Attach a POP to an object” on page 92
• “Find where a POP is attached” on page 92
• “Detach a POP from a protected object” on page 92

ACL policies operate in a similar way to protected object policies (POPs). Both POPs and ACL policies are inherited in the same way. See “Managing an access control list” on page 76 for information about how ACL policies operate.

Create a POP

To create a POP using the Web Portal Manager or the pdadmin command line utility, log in to the desired domain as a domain administrator.

Web Portal Manager
To create a POP using Web Portal Manager:
1. Log in to the domain.
2. Click POP ➔ Create POP to display the Create POP window.
3. Type a POP Name (for example, pop1). This field is required.
4. Optionally, type a Description of the POP.
5. Select one or more check boxes for the appropriate audit levels. The **Audit Level** is the level of auditing that applies when a resource, to which this POP is attached, is accessed. You can select more than one audit level. The audit levels are as follows:

   - **Permit**
     Audits all requests on a protected object that result in successful access.
   - **Deny**
     Audits all requests on a protected object that result in denial of access.
   - **Error**
     Audits all internally generated error messages resulting from a denial of access to the protected object.
   - **Admin**
     Audits not used by Tivoli Access Manager. However, this option can be used by custom applications.

   For more information, refer to "Audit level attribute" on page 93.

6. Select the check box for **Warn Only On Policy Violation** to enable the warning mode attributes. A warning mode attribute indicates whether a policy violation that is related to a resource results in denial of access or in an audited failure. An **audited failure** is an access attempt to a resource, to which a POP applies, that results in the access being audited, not denied.

   For more information, refer to "Warning mode attribute" on page 93.

7. Select a type of **Quality of Protection**. The level of protection that applies when a resource, to which this POP is attached, is accessed. The quality of protection choices are:

   - **None**
     Requires no quality of protection.
   - **Integrity**
     Uses some mechanism to ensure that the data has not changed.
   - **Privacy**
     Requires data encryption for Secure Sockets Layer (SSL).

   For more information, refer to "Quality of protection POP policy" on page 99.

8. For **Time of Day Access**, specify the days and times of the day that the resource can be accessed.

   - Select the check boxes for the days of the week that the resource can be accessed.
   - Select either **All Day** or **Between hours of** for the access times that the resource can be accessed on the selected days.
   - If you select **Between hours of**, you must also specify the **Start time** and **End time**.
   - If you select **Between hours of**, you must also specify the **Local Time** or **UTC Time** (coordinated universal time).

   For more information, refer to "Time-of-day attribute" on page 94.

9. Click **Create**. If successful, you will see a message confirming the POP has been created.

10. Click **Create Another** if you want to create another POP, or else click **Done**.
**pdadmin**
To create a POP in the domain using the `pdadmin` utility, log in to the domain and use the `pdadmin pop create` command. For example, to create a POP named `poptest1`, enter the following:

```
padmin sec_master> pop create poptest1
```

The new POP contains the following default settings:

```
padmin sec_master> pop show poptest1
  Protected object policy: poptest1
  Description:               
  Warning: no
  Audit level: none          
  Quality of protection: none
  Time of day access: sun, mon, tue, wed, thu, fri, sat: 
    anytime:local
  IP Endpoint Authentication Method Policy
    Any Other Network 0
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*.

**List POPs**
To list all POPs using the Web Portal Manager or the `pdadmin` command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**
To view a list of all POPs using Web Portal Manager:
1. Log in to the domain.
2. Click **POP > List POP** to display the Manage POPs window.
   - All POPs for the domain will be listed as links.

**pdadmin**
To list all POPs in the domain using the `pdadmin` utility, log in to the domain and use the `pdadmin pop list` command.

```
padmin sec_master> pop list
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*.

**Delete a POP**
To delete a POP using the Web Portal Manager or the `pdadmin` command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**
To delete a POP using Web Portal Manager:
1. Log in to the domain.
2. Click **POP > List POP** to display the Manage POPs window.
3. Select one or more check boxes for the POPs that you want to delete.
4. Click **Delete**.
5. Confirm the deletion on the Delete Pop window by clicking **Delete**.

**pdadmin**
To delete a POP in the domain using the `pdadmin` utility, log in to the domain and use the `pdadmin pop delete` command. For example, to delete the POP named `poptest2`, enter the following:

```
padmin sec_master> pop delete poptest2
```
pdadmin sec_master> pop delete poptest2

For more information, see the IBM Tivoli Access Manager for e-business Command Reference.

Modify a POP
To modify a POP using the Web Portal Manager or the pdadmin command line utility, log in to the desired domain as a domain administrator.

Web Portal Manager
To modify a POP using Web Portal Manager:
1. Log in to the domain.
2. Click POP → List POP to display the Manage POPs window.
3. Click on the link for the POP (for example, poptest1 to display the POP Properties window.
4. At the General tab, change the information pertaining to the POP, as needed. For example, change the description from Test POP to Test 1 for POP, and then click Apply.
   For online help for any Web Portal Manager window, click the question mark to open a separate help window.
5. Click the Attach tab to view a list of, attach the specified POP to, or detach the POP from, protected objects.
6. Click the IP Auth tab to view a list of, create, or delete IP authentication.
7. Click the Extended Attributes tab to view a list of, create or delete extended attributes.

pdadmin
To modify a POP in the domain by using the pdadmin utility, log in to the domain, and use the pdadmin pop modify commands. For example, to modify the POP named poptest1, enter the following:
padmin sec_master> pop modify poptest1 set description "Test 1 for POP"

Note: Always enclose the description with double quotation marks when you use more than one word.

To show the modifications to the POP use the pdadmin pop show command.
padmin sec_master> pop show poptest1

Protected object policy: poptest1
Description: Test 1 for POP
Warning: no
Audit level: none
Quality of protection: none
Time of day access: sun, mon, tue, wed, thu, fri, sat: anytime:local
IP Endpoint Authentication Method Policy
   Any Other Network 0

For more information, see the IBM Tivoli Access Manager for e-business Command Reference.

Applying POP attributes to protected objects
POP policies are applied to objects in the same manner as ACL policies.
**Attach a POP to an object**
To attach a POP to an object using the Web Portal Manager or the `pdadmin` command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**
To attach a POP to an object using Web Portal Manager:
1. Log in to the domain.
2. Click **POP → List POP** to display the Manage POps window.
3. Click the link for the POP.
4. From the POP Properties window, click the Attach tab.
5. Click Attach to display the Attach POP window.
6. Type the Protected Object Path for the protected object to which the POP will be attached. Express the path as the full path name (for example, `/WebSEAL/serverA/index.html`).
7. Click Attach. If successful, the protected object is added to the list at the POP Properties–Attach window.

**pdadmin**
To attach a POP to a protected object in the domain by using the `pdadmin` utility, log in to the domain, and use the `pdadmin pop attach` command. For example, to attach a POP named test to a protected object named `/WebSEAL/serverA/index.html` enter the following:

```
pdadmin sec_master> pop attach /WebSEAL/serverA/index.html test
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*.

**Find where a POP is attached**
To find where a POP is attached using the Web Portal Manager or the `pdadmin` command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**
To find where a POP is attached using Web Portal Manager:
1. Log in to the domain.
2. Click **ACL → List POP**. A list of POP names is displayed. Each POP name is a link that you can click to display the POP properties window.
3. Click the Attach tab.

**pdadmin**
To find where a POP is attached in the domain using the `pdadmin` utility, log in to the domain and use the `pdadmin pop find` command. For example, to find where the POP named test is attached, enter the following:

```
pdadmin sec_master> pop find test
```

```
/WebSEAL/serverA/index.html
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*.

**Detach a POP from a protected object**
To detach a POP from a protected object using the Web Portal Manager or the `pdadmin` command line utility, log in to the desired domain as a domain administrator.
**Web Portal Manager**

To detach a POP from a protected object using Web Portal Manager:

1. Log in to the domain.
2. Click **Object Space** → **Browse** to display the Browse Object Space window.
3. Click the link for the POP.
4. From the POP Properties window, click the **Attach** tab.
5. Select one or more check boxes for the protected objects from which you want to detach the POP.
6. Click **Detach** to display the Detach POP from Object window where you are prompted to confirm or cancel the detachment.

**pdadmin**

To detach a POP from a protected resource in the domain by using the **pdadmin** utility, log in to the domain, and use the **pdadmin pop detach** commands. For example, to detach the POP from the protected object named `/WebSEAL/serverA/index.html`, enter the following:

```
padmin sec_master> pop detach /WebSEAL/serverA/index.html
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*.

### Configuring the POP attributes

POP attributes impose access conditions on an object based on the time of the access and to indicate whether the access request should be audited.

#### Warning mode attribute

The purpose of the warning attribute is to allow a security administrator to debug or troubleshoot the accuracy of the authorization policy set on the protected object space.

When you set the warning attribute to yes, any action is possible by any user on the object where the POP is attached. Any access to an object is permitted even if the security policy attached to the object is set to deny this access.

Audit records are generated that capture the results of all security policies with warning mode set throughout the object space. The audit log shows the outcome of an authorization decision as it would have been made if the warning attribute has been set to no. Therefore, the administrator can determine if policy is set and enforced correctly.

For example:

```
padmin sec_master> pop modify test set warning yes
```

For more information about the **pdadmin pop** commands, see *IBM Tivoli Access Manager for e-business Command Reference*.

#### Audit level attribute

The POP audit level has the expanded ability to specify a level of auditing. For example, if auditing is set to record unsuccessful events, you can use the results to detect an unusual number of failed access attempts on a particular resource.
Auditing records are written in a standard Extensible Markup Language (XML) format that allows easy parsing to extract whatever information is required. See “Overview of auditing” on page 219.

For example:

```
pdadmin sec_master> pop modify test set audit-level permit,deny
```

<table>
<thead>
<tr>
<th>Audit-Level-List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>permit</td>
</tr>
<tr>
<td>deny</td>
</tr>
<tr>
<td>error</td>
</tr>
</tbody>
</table>

You can apply any combination of these three values. Use a comma as a separator character when you specify more than one value.

For more information about the `pdadmin pop` commands, see IBM Tivoli Access Manager for e-business Command Reference.

**Time-of-day attribute**

The time-of-day (TOD) POP attribute allows you to place specific day and time conditions on the access to a protected object. This type of condition might be useful to limit access to information that regularly requires periods of inactivity for modification and updates.

```
pdadmin sec_master> pop modify pop_name set tod-access
time_of_day_string
```

The time-of-day-string argument includes a day-range and a time-range and uses the following format:

```
{anyday|weekday|day_list}:{anytime|time_spec-time_spec}[:{utc|local}]
```

The `day_list` variable can be any combination of the following:

- mon, tue, wed, thu, fri, sat, sun

The `time_spec` range variable must be expressed (using 24 hour time) in the following format:

```
hhmm-hhmm
```

For example:

```
0700-1945
```

The optional time zone `[:{utc|local}]` for the server (not the client) is `local` by default.

For example:

```
pdadmin sec_master> pop modify test set tod-access mon, tue, fri:1315-1730
```

For more information about the `pdadmin pop` commands, see IBM Tivoli Access Manager for e-business Command Reference.
Network-based authorization POP policy

The network-based authorization POP policy makes it possible to control access to objects based on the IP address of the user. You can use this functionality to prevent specific IP addresses (or IP address ranges) from accessing any resources in your domain.

You can also apply step-up authentication configuration to this policy and require a specific authentication method for each specified IP address range.

Network-based authorization policy is set in the IP endpoint authentication method attribute of a POP policy. You must specify two parts for this attribute:

- Authentication levels
  
  For more information on authentication levels, see "Authentication strength POP policy (step-up)" on page 97.

- Allowed networks

Specifying IP addresses and ranges

The popadmin pop modify set ipauth add command specifies both the network (or network range) and the required authentication level in the IP endpoint authentication method attribute.

For more information about the popadmin pop modify set ipauth add command, see the IBM Tivoli Access Manager for e-business Command Reference.

The configured authentication levels are linked to IP address ranges. This method is intended to provide flexibility. If filtering users by IP address is not important, you can set a single entry for anyothernw (any other network). This setting affects all accessing users, regardless of IP address, and requires them to authenticate at the specified level.

Conversely, if you want to ignore the authentication level and want to allow or deny access based only on IP address, you can use level 0 for ranges that you want to allow in and forbidden for ranges you want to deny.

The anyothernw entry is used as a network range that matches any network not otherwise specified in the POP. This method is used to create a default entry that could either deny all unmatched IP addresses or allow access to anyone who meets the authentication level requirement.

By default, anyothernw appears in a POP with an authentication level index of 0. The entry appears as Any Other Network in the pop show command:

```
padmin sec_master> pop show test
Protected object policy: test
Description: Test POP
Warning: no
Audit level: none
Quality of protection: none
Time of day access: sun, mon, tue, wed, thu, fri, sat:
  anytime:local
IP Endpoint Authentication Method Policy
Any Other Network 0
```

For more information about the popadmin pop modify set ipauth anyothernw command, the IBM Tivoli Access Manager for e-business Command Reference.
Granting access to identifies
This example grants access to identities from IP addresses that begin with 9.
pdadmin sec_master> pop modify test set ipauth add 9.0.0.0 255.0.0.0

This example grants access to identities from IP address 9.1.2.3.
pdadmin sec_master> pop modify test set ipauth add 9.1.2.3 255.255.255.255

This example prevents all users (other than those specified as in the examples
above) from accessing the object:
pdadmin sec_master> pop modify test set ipauth anyothernw forbidden

For more information about the pdadmin pop modify set ipauth add and the
pdadmin pop modify set ipauth forbidden commands, see the IBM Tivoli Access
Manager for e-business Command Reference.

Disabling network-based authorization by IP address
This example removes access to identities from IP addresses that begin with 9.
pdadmin sec_master> pop modify test set ipauth remove 9.0.0.0 255.0.0.0

For more information about the pdadmin pop modify set ipauth remove
command, see the IBM Tivoli Access Manager for e-business Command Reference.

Network-based authorization algorithm
The authorization engine uses the following algorithm to process the conditions in
a POP:
1. Check ACL permissions.

   Note: There is an ACL policy Bypass (B) permission that overrides POP
   authorization conditions on an object. This permission should be used
   only by a high-level administrator who needs full access to the protected
   object space all the time.
2. Check to see if a rule is attached to the object, then verify that all of the access
decision information (ADI) is present for the coming rule evaluation. If it is not,
then find it by querying one of the available sources.
3. Check the IP endpoint authentication method policy on the POP.
4. Check time-of-day policy on the POP.
5. Check the audit level policy on the POP.
6. Check the rule policy if a rule is attached to the object.
7. If an external authorization service (EAS) operation or POP trigger applies to
   this access decision, then invoke the EAS that apply.

Network-based authorization notes and limitations
The IP address used by the resource manager for enforcing the network-based
authorization policy should be the IP address of the originator of the connection. If
your network topology uses proxies, the address that appears to the resource
manager might be the IP address of the proxy server.

In this case, the resource manager is not able to definitively identify the true client
IP address. You must be careful when setting a network-based authorization policy
that network clients can directly connect to the resource manager.
Authentication strength POP policy (step-up)

You can use protected object policies (POP) to enforce certain access conditions on specific resources. The authentication strength POP policy makes it possible to control access to objects based on authentication method.

You can use this functionality, sometimes known as step-up authentication, to ensure that users accessing more sensitive resources use a stronger authentication mechanism. You might want this condition because of the greater threat of improper access to certain resources.

For example, you can provide greater security to a junctioned region of the protected object space by applying a step-up POP policy that requires a stronger level of authentication than the client used when initially entering the domain.

Authentication strength policy is set in the IP endpoint authentication method attribute of a POP policy.

Configuring levels for step-up authentication

The first step in configuring authentication-specific access is to configure the supported authentication methods and determine the order in which these authentication methods should be considered stronger.

Any client accessing a resource manager has an authentication level, such as “unauthenticated” or “password”, which indicates the method by which the client last authenticated with the resource manager.

In some situations, it might be necessary to enforce minimum safe levels of authentication required to access certain resources. For example, in one environment, authentication by token pass code might be considered more secure than authentication by user name and password. Another environment might require different standards.

Rather than forcing clients to restart their sessions with the resource manager when they do not meet the required level of authentication, the step-up authentication mechanism provides clients a second chance to re-authenticate using the required method (level).

Step-up authentication allows resource managers to control the method in which users access a protected resource. If step-up authentication is required because the user has not authenticated with the sufficient method, then the access decision is still permitted by the authorization engine but the resource manager is presented with a required authentication level as an output of the authorization decision. The resource manager can then decide how to further authenticate the user so as to gain the required level of authentication needed for the user to access the object.

How a particular authentication method is mapped to an authentication level is entirely determined by the resource manager application. For all cases, the absolute minimum acceptable method of authentication should be set as level 0 with more secure methods being mapped to integral numbers in ascending order (1..x) from there.
Applying step-up authentication policy

Step-up authentication is implemented through a POP policy placed on the objects requiring authentication-sensitive authorization. You use the IP endpoint authentication method attribute of a POP policy.

The `pdadmin pop modify set ipauth` command specifies both the allowed networks and the required authentication level in the IP endpoint authentication method attribute.

The configured authentication levels can be linked to IP address ranges. This method is intended to provide management flexibility. If filtering users by IP address is not important, you can set a single entry for `anyothernw` (any other network). This setting affects all accessing users, regardless of IP address, and require them to authenticate at the specified level. This is the most common method for implementing step-up authentication.

The `anyothernw` entry is used as a network range that matches any network not otherwise specified in the POP. This method used to create a default entry that could either deny all unmatched IP addresses or allow anyone access who can meet the authentication level requirement.

By default, `anyothernw` appears in a POP with an authentication level index of 0. The entry appears as Any Other Network in the `pdadmin pop show` command:

```
padmin sec_master> pop show test
   Protected object policy: test
   Description: Test POP
   Warning: no
   Audit level: none
   Quality of protection: none
   Time of day access: sun, mon, tue, wed, thu, fri, sat:
      anytime:local
   IP Endpoint Authentication Method Policy
      Any Other Network0
```

For more information about the `pdadmin pop modify set ipauth` command, see the *IBM Tivoli Access Manager for e-business Command Reference*.

Distinguishing step-up from multi-factor authentication

Tivoli Access Manager step-up authentication and multi-factor authentication are two different and distinct mechanisms for controlling access to resources. Tivoli Access Manager provides only step-up authentication functionality, as described in this chapter.

Multi-factor authentication forces a user to authenticate using two or more levels of authentication. For example, the access control on a protected resource can require that the user authenticate with both user name/password and user name/token passcode.

Tivoli Access Manager step-up authentication relies on a pre-configured hierarchy of authentication levels and enforces a specific level of authentication according to the policy set on a resource. Step-up authentication does not force the user to authenticate using multiple levels of authentication to access any given resource. Instead, step-up authentication requires the user to authenticate at a level at least as high as that required by the policy protecting the resource.
A step-up authentication example is as shown:

The following are configured authentication levels:
- authentication level 1 = user name/password
- authentication level 2 = user name/token passcode

The following object is protected by a POP requiring authentication level 1:
/WebSEAL/hostA/junction

The following object is protected by a POP requiring authentication level 2:
/WebSEAL/hostA/junction/applicationA

Under step-up authentication, user name/password (level 1) authentication is required to access /WebSEAL/hostA/junction.

However, user name/token passcode (level 2) authentication is required to access /WebSEAL/hostA/junction/applicationA. If the user is currently logged in with a user name and password, a prompt appears requesting user name and token passcode information (the step-up). However, if the user initially logs in to WebSEAL using a user name and a token passcode, access to applicationA is immediate (assuming a successful ACL check).

Multi-factor authentication would require both level 1 and level 2 authentication for access to applicationA.

### Quality of protection POP policy

The quality of protection POP attribute allows you to specify what level of data protection is required when performing an operation on an object.

The quality of protection POP attribute permits a single transaction where the yes response to the ACL decision also includes the required quality of protection level. If the resource manager cannot guarantee the required level of protection, the request is denied.

**Syntax:**

```
padmin sec_master> pop modify pop-name set qop {none|integrity|privacy}
```

<table>
<thead>
<tr>
<th>QOP level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privacy</td>
<td>Data encryption is required for Secure Sockets Layer (SSL).</td>
</tr>
<tr>
<td>Integrity</td>
<td>Use some mechanism to ensure that the data has not changed.</td>
</tr>
</tbody>
</table>

For example:
```
padmin sec_master> pop modify test set qop privacy
```
Chapter 10. Authorization rules management

This chapter provides detailed information about Tivoli Access Manager authorization rules (AuthzRules). Authorization rules are conditions contained in an authorization policy that are used to make access decisions based on attributes such as user, application, and environment context.

This chapter contains the following sections:

- “Authorization rules overview”
- “Access decision information”
- “Authorization rule language” on page 104
- “Authorization rules evaluator” on page 110
- “Example rules” on page 113
- “Methods of providing ADI to the rules evaluator” on page 116
- “Reason codes for rule failures” on page 117
- “Configuration file and initialization attributes” on page 118
- “Managing an authorization rule” on page 120

Authorization rules overview

Authorization rules are defined to specify conditions that must be met before access to a protected object is permitted. A rule is created using a number of Boolean conditions that are based on data supplied to the authorization engine within the user credential, from the resource manager application, or from the encompassing business environment. The language of an authorization rule allows customers to work with complex, structured data by examining the values in that data and making informed access decisions. This information can be defined statically within the system or defined during the course of a business process. Rules can also be used to implement extensible, attribute-based, authorization policy by using attributes within the business environment or attributes from trusted external sources.

A Tivoli Access Manager authorization rule is a policy type similar to an access control list (ACL) or a protected object policy (POP). The rule is stored as a text rule within a rule policy object and is attached to a protected object in the same way and with similar constraints as ACLs and POPs.

Access decision information

The data and attributes that are used in rule conditions collectively are called access decision information (ADI). Authorization API attributes, which are name and value pairs, form the basis of all ADI that can be referenced in a rule or presented to the authorization engine.

Sources for retrieving ADI

There are four sources from which the authorization engine can gather ADI for use when evaluating a rule:

- User credential entitlements
- Application context information passed in by the Tivoli Access Manager resource manager
- Tivoli Access Manager authorization engine context
- Dynamic ADI retrieval entitlement services

**User credential entitlements**

Additional entitlements data can be inserted as name and value attribute pairs (referred to as tag-value) into the client credential by a Tivoli Access Manager authorization client during the user authentication process or at any time during the process of the transaction. For example, Tivoli Access Manager can be configured using tag-value support to gather entitlements at the time that a user is authenticated. You can configure entitlement services to run during credential acquisition, collect entitlements data, and then append the data to the credential. Tivoli Access Manager provides a credential attributes entitlement service that retrieves entitlements data from the user registry. Or, you can define your own entitlement services. For more information about defining entitlement services and tag-value support, refer to *IBM Tivoli Access Manager for e-business Authorization C API Developer Reference*.

Any attribute added to the user credential can be used as ADI in a rule definition. There are also attributes that are built into the Tivoli Access Manager user credential when it is created by the authorization engine. Just like attributes that can be added to the credential by the resource manager, the built-in credential attributes can be used in authorization rules. The built-in credential attributes include items of information, such as the user name (or the principal UUID) and the groups (or the group UUID) of which the user is a member.

See the *IBM Tivoli Access Manager for e-business Authorization C API Developer Reference* for a table of valid credential attribute names. All credential attribute names begin with `azn_cred_` (for example, `azn_cred_principal_uuid`). This table lists attribute names available within the Tivoli Access Manager authenticated user credential, their value, and a description. Many attributes in this table are also available in an unauthenticated user credential but those related to a user’s identity are not. For example, attributes such as the user name, principal UUID, group name, and group UUID, as well as the LDAP DN for LDAP configurations are not available in an unauthenticated credential. Keep in mind that, when developing rules that use these particular attributes, the authorization engine requires that all ADI be present before a rule can be evaluated. If the ADI is not available, the authorization decision will be returned with an error status. Requiring that the user be authenticated before accessing the protected object with such a rule attached ensures that the authenticated credential information is available. This requirement can be achieved using an ACL entry on the object that requires authenticated access.

**Application context information**

Authorization rules might require application context information to complete an evaluation. Context information include information that is not an entitlement but is specific to the current transaction or operation. An example is a transaction amount, such as purchase price or transfer amount. This information is passed to the decision through the `app_context` attribute list of the `azn_decision_access_allowed_ext()` call. Tivoli Access Manager WebSEAL also uses this mechanism to pass the values of certain HTML tags and HTML request data (from a get or post request) into the access decision for use in a rule evaluation.
Authorization engine context information
Authorization engine context information is provided automatically by the authorization engine, if required, before the authorization rule is evaluated. The ADI provided by the authorization engine includes the name of the protected object that is the target of the access decision and the string of operations that the requesting user wants to perform on the protected object.

The following attribute names are reserved for these data items:
- `azn_engine_target_resource`
- `azn_engine_requested_actions`

Dynamic ADI retrieval entitlement services
The final source for retrieving ADI is the dynamic ADI retrieval entitlement service. This class of authorization entitlement services is designed to retrieve ADI from an external source. These services can be developed to retrieve ADI from an enterprise database containing employee, customer, partner or inventory information. The dynamic ADI retrieval service is called to retrieve ADI at the time that the access decision is being made. Calling both at the same time has the benefit of being able to retrieve volatile data, such as quotas, at a time when its value is most current.

The Tivoli Access Manager Attribute Retrieval Service (AMWebARS) is an example of a service that can retrieve ADI from external sources. AMWebARS is the official package name for a Tivoli Access Manager J2EE Web service that implements a dynamic ADI retrieval service. To facilitate communication between the resource manager, which is invoking the rules engine, and AMWebARS, which is performed using SOAP over HTTP, the Access Manager runtime environment (PDRTE package) provides an authorization entitlement service called `azn_ent_amwebars`.

Refer to the IBM Tivoli Access Manager for e-business Authorization C API Developer Reference for more information on developing and using dynamic ADI retrieval entitlement services to fetch ADI at the time the rule is evaluated. Also, refer to the IBM Tivoli Access Manager for e-business Administration C API Developer Reference for an in-depth discussion of the attribute lists and their format and of the authorization API that are used to manipulate them. For more information on the constraints and format for ADI, refer to "Format and constraints of rules" on page 111.

Volatile versus non-violatilere data
In general, the source for any particular piece of ADI depends largely on what the data is. The most important question is whether the data is volatile. For example, is it possible for the data to change during the lifetime of the user’s session and, if so, is it important to use the most up-to-date information when it does? Volatile data should be retrieved using a dynamic ADI retrieval service unless the resource manager application can provide this data.

Application-specific data that is non-violatile and not user-specific is provided by the resource manager application. Data that is non-violatile and user-specific is loaded into the user credential when the user is authenticated and is kept with the credential for the lifetime of the user’s session.

The set of data provided by the authorization engine, including the target protected object and permissions, is fixed and cannot be changed.
Authorization rule language

eXtensible Stylesheet Language (XSL) is the language used to specify rules and eXtensible Markup Language (XML) is the language used for the data that forms an input to the rules. The combination of XML and XSL provides a platform independent way to express both the inputs to the rules evaluator and the rules themselves.

XML also provides the ability to express complex data types in a structured and standard manner in text format. This text format allows rules for processing the XML data to be written without having to cater to platform and programming language specifics.

XSL is a functional style sheet language that can be used to perform simple tasks or complex tasks depending on your needs. XSL possesses an inherent ability to analyze and evaluate XML data, which is becoming the standard for data representation in e-business models. XSL is built on other XML-based standards such as XPath, which is the expression language at the core of an authorization rule.

To implement rules-based authorization policy, it is necessary to impose a number of constraints on the XSL rules, including the requirements that the output of the rule evaluation be simple text and that the output conforms to one of a known set of result strings. For more information on the format and constraints of authorization rules refer to “Format and constraints of rules” on page 111.

It is also necessary to impose constraints on the XML input document that is built as input to the rule evaluation. The ADI XML document model enables the authorization engine to detect when ADI is missing and when it needs to be requested from the resource manager or an external entity through the dynamic ADI retrieval service interface.

ADI XML document model

The ADI XML document model (or ADI XML model) is a set of restrictions placed on the XSL/XML model by the authorization rules implementation to enable the interface to be simple and yet functional for authorization purposes. The model constrains the authorization rules to function within a predetermined XML document format with the same top-level XML document element for all rules. The XML ADI that is imported by the rules evaluator from credential attributes, from application context, or from other data sources must be inserted into this XML document before authorization rules can use the data. Similarly to simplify the process of defining rules, the authorization rules must operate within the confines of the ADI XML model. The ADI XML model requires the XML document to contain the following top-level XML element into which all target ADI for a particular rule evaluation is inserted. The XMLADI element is created automatically as part of the rule evaluation process by the authorization engine.

```
<XMLADI>
<!-- XML formatted ADI are inserted here. -->
</XMLADI>
```

As a result of this restriction, the XPath to the data used in an authorization rule must include the prefix /XMLADI to access a particular data element within the model. For example, if an ADI item of JohnSmith is added to the document to access the fields of JohnSmith within the ADI XML document, you will have to specify the XPath /XMLADI/JohnSmith to access the data contained in the XML object JohnSmith.
An XPath is the path to a particular child element within the hierarchy of a structured XML data object. Much like a directory path on a hard drive is used to access a specific file, an XPath designation starts from the root of the document (in this case /XMLADI) and traces a path from this root down through its child elements to the specific element that is being referenced. For example, using the example entitlement JohnSmith in the "XML entitlement example" on page 107 as a reference, the JohnSmith XML object has a child element called CreditCard. The child elements of the CreditCard element are attributes which are common to most credit cards. To access Balance under the CreditCard element of JohnSmith, you would use the XPath:

"/XMLADI/JohnSmith/CreditCard/Balance"

XPaths like this example are the means by which authorization rules access the ADI data values that are needed to make attribute-based authorization decisions.

Because all data elements are restricted to work within the ADI XML model, the authorization rules must also be restricted to operate on or match XPaths within the model. Therefore, XSL template match statements are also restricted to matching XPaths starting from /XMLADI within the ADI XML document. For additional information, see "Format and constraints of rules" on page 111.

Containers and XML ADI container names
When data is requested from a resource manager, the granularity of any XML data returned is at the level of a single container of information. The container is normally also the smallest data element (for example, elements that might be considered for billing purposes). This convention has been adopted for the ADI XML model as well. The ADI that is used in authorization rules is also defined and manipulated as containers of XML data. For example, the JohnSmith XML object defined in "XML entitlement example" on page 107 is an example of an ADI container.

To this end, the top-most element in the definition of an item of ADI is referred to as the container name of that item of ADI. When defining an authorization rule, the XPath to the XML definition of data in any ADI container must always be referenced using the name of the container as the first element following /XMLADI in the XPath specification for the data element.

Returning to the example ADI item JohnSmith, you can assume there is a container received from the data provider named JohnSmith. To access any element within the JohnSmith container, the XPath specification must be prefixed with JohnSmith. For example, JohnSmith/CreditCard/AccountNumber refers to the AccountNumber value. To access this information from within an authorization rule, this XPath must also be prefixed by the top-level element of the XML target ADI input document, which is XMLADI (for example, /XMLADI/JohnSmith/CreditCard/AccountNumber). However, both of these XPaths are valid when used in an authorization rule due to the default template match statement that is added to all authorization rules that do not explicitly include one. Because the default template match statement matches the ADI XML document from /XMLADI, JohnSmith can be referred to either with a relative reference or with an absolute reference that is prefixed with /XMLADI. For additional information, see "Format and constraints of rules" on page 111.

Limitations of container names
One restriction imposed by the ADI XML document model is that each item of ADI that can be consumed by the rules evaluator must have a unique container name that cannot be confused with containers provided by other entitlements data providers. For example, if two different data providers provide a data item called
TxInfo, there will be no way for the rules evaluator to know which provider it should make a request to in order to get this item of data. To help differentiate items of ADI with the same name, XML provides the ability to define namespaces for data. The namespace ID of the namespace can then be used to differentiate one ADI element from another. In case of TxInfo, we could define a namespace `companyA` and reference this instance of ADI with `companyA:TxInfo`. For more information about namespace definitions, see "Defining an XML namespace" on page 108.

This restriction on container naming among data providers is not specifically enforced by the authorization engine. On the contrary, if the engine encounters multiple instances of the same item of ADI (for example, TxInfo), it will simply add them all to the ADI XML document for use in the evaluation. In the ADI XML document, there can be two items of ADI data with the same container name within the ADI XML input document. The assumption is then made that they are structured in the exact same way. For example, a particular application request might involve a number of individual transactions each with its own transaction amount. An authorization rule can be formulated to add all of these items together and compare the sum of the items to a predefined total transactions limit or to a per-transaction limit using an XSL node select statement. "Example 3" on page 115 in the "Example rules" on page 113 section in this chapter shows an example rule that sums multiple transaction elements in this way and even counts the number of instances of a particular ADI element.

**XML access decision information**

By default, the rule evaluator automatically transforms into XML format any name/value pair attributes passed to it by the calling application that have been identified as target access decision information (ADI) for the current evaluation. When transforming the attribute to XML, the attribute name is used as the container name of the XML data item and the attribute value is converted into an XML value. The container name of an item of ADI equates to the XML element name in the XML definition. For example, the following XML data is generated for attribute name VPS_CREDIT_CARD with a string attribute value of 5517 3394 8324 0965:

```xml
<VPS_CREDIT_CARD>5517 3394 8324 0965</VPS_CREDIT_CARD>
```

The container name and XML element name in this case is VPS_CREDIT_CARD. The graphical user interface, the command line interface, and the Tivoli Access Manager authorization API attribute list interfaces do not permit the administrator to define rules that contain invalid XML container names.

If the application passes entitlements or application context that have already been formatted as XML for an access decision, the authorization rules evaluator will expect the data to be of type `azn_string_t` and will expect the format of the string to be XML. The attribute name must match the container name of the XML data item. If the names do not match, the evaluator does not evaluate the rule correctly.

The evaluator identifies XML format data by locating the `<` character at the beginning of the attribute value. If the attribute value does not begin with a `<` character, the data is not considered to be an XML data item and the evaluator attempts to convert the data item to XML format automatically. This means of identification is used only on attributes or application context identified as target ADI for the access decision. Therefore, non-XML attribute values starting with a `<` character cannot be used by the application and results in an error status return.
from the authorization decision. If the data is not correct XML, the XSL processor fails and returns an error to denote the failure.

Data items that should be defined in XML must be entirely defined in XML and should not rely on the translation mechanism for non-XML items to generate the appropriate XML element name automatically. For example, to define an attribute to contain the XML definition of MY_CREDIT_CARD_NUM, you must add an attribute with the attribute name MY_CREDIT_CARD_NUM. The attribute value for MY_CREDIT_CARD_NUM is the following:

<MY_CREDIT_CARD_NUM>500</MY_CREDIT_CARD_NUM>

By defining the XML element completely as opposed to only defining its value, XML attributes can be added to the element definition without affecting the name by which the ADI is referred to when talking with data providers.

For example, in the following definition of the XML item MY_CREDIT_CARD_NUM we have defined an XML attribute CardType with the value of "visa". XML attributes are defined in the element start tag of the element to which they apply. Functionally XML attributes are equivalent to any other first-level child element of the XML object. To reference the attribute CardType, the required XPath would be:

/XMLADI/MY_CREDIT_CARD_NUM/CardType

XML attributes should not be confused with the authorization API attributes and attribute lists that are used to carry data into and out of the authorization process.

<MY_CREDIT_CARD_NUM CardType="visa">
  5517 3394 8324 0965
</MY_CREDIT_CARD_NUM>

The ability to add XML attributes to an element definition is particularly useful when it comes to defining a namespace for the data item. For more information on XML namespaces, see "Defining an XML namespace" on page 108.

If the ADI attribute contains multiple attribute values (string, XML, or any combination thereof), the evaluator will convert each attribute value as a separate instance of ADI. For example, for an attribute named Tx0data with attribute values 100 and 500, the evaluator inserts the following XML item declarations into the ADI XML document:

<TxData>100</TxData>
<TxData>500</TxData>

The policy administrator can then design an authorization rule that uses XSL language node selection statements to work with the two values independently or to add the values and compare the sum total with some predefined limit. If Tx0data is compared to a value of some kind, it is treated as a node set comparison and each Tx0data value is compared to the data in turn with success being indicated if any of the Tx0data values equal the target data. Node set comparisons have slightly different behavior than expected when using the "!=" operator, in most cases, and the not() function, needs to be used instead. For an example of where to use "!=" and not() when comparing a node set, see "Example 3" on page 115.

**XML entitlement example**

The following example is an ADI XML document that might be passed to the XSL processor from the rules evaluator during the evaluation of an authorization rule.
The document contains two containers: JohnSmith and AmountReqd. The attribute value of the container JohnSmith is defined in XML. The AmountReqd container is translated to XML from an incoming string application context attribute. The container JohnSmith is an entitlement and the container AmountReqd is an item of transaction context.

The authorization rules evaluator automatically encompasses all of the data under the XML top-level node declaration XMLADI when the ADI XML document is created, so this top-level element has been added for clarity.

The XML document that is passed to the evaluation routines by the authorization rules evaluator is as follows:

```
<XMLADI>
  <JohnSmith>
    <CreditCard>
      <AcctNumber>0123456776543210</AcctNumber>
      <Limit>10000.00</Limit>
      <Balance>2000.00</Balance>
    </CreditCard>
    <MileagePlus>
      <MemberStatus>100k</MemberStatus>
      <CardNumber>12345678</CardNumber>
    </MileagePlus>
  </JohnSmith>
  <AmountReqd>500.00</AmountReqd>
</XMLADI>
```

When referencing a particular ADI item within the XMLADI document available to a rule, the XPath path specifier can begin from the container name of the XML element, for example, JohnSmith, as the default template rule matches the /XMLADI element automatically. If the callers want to specify their own template match statement explicitly, they will be able to do so.

In this example, the ADI container names are JohnSmith and AmountReqd. For additional information, see “Format and constraints of rules” on page 111.

**Defining an XML namespace**

XML namespaces are used to differentiate between XML items with the same name or to group XML data of the same type or function together. The same principles can be used with ADI that is defined for use with authorization rules. For example, a customer database and a product inventory database might both define ADI called name that could be used in authorization rules. By defining an XML namespace with the namespace ID item, you can differentiate between the two instances of name by calling the ADI from the product database item:name. This example provides a namespace definition for the item namespace:

```
xmlns:item="http://mycompany/namespaces/items"
```

where xmlns is a standard XML attribute name and item is the namespace ID chosen for the namespace. The URI following the = is used to distinguish one namespace ID from another.

This namespace declaration associates the namespace ID item with the URI string:

```
http://mycompany/namespaces/items
```
The value of the URI string is of no consequence to the XML and XSL processors but it must be unique. Unlike the XML and XSL processors, the Tivoli Access Manager authorization engine does not permit two namespace IDs to be assigned the same URI value as it uses the URI to uniquely identify the namespaces itself. Defining two namespaces with the same URI results in an initialization error, the authorization application will not start and an error will be logged to the application’s error log. The source from which the item name is to be obtained must be aware of this relationship. The source must be able to make the connection between the item:name requested by the authorization engine and the name data stored in the product database. The source must also be able to provide this data to the authorization engine in an attribute called item:name when it is needed. For example, a dynamic ADI retrieval service must understand that, when it is asked for item:name, it should fetch the required value by looking for name in the product database. The service needs to return the data to the authorization engine in an attribute called item:name. When an application uses namespaces to differentiate or aggregate ADI items, it is required to define the namespace for both the XML and XSL processors.

To define a namespace for the XSL processor add the namespace definition to the xsl-stylesheet-prolog configuration file entry discussed in "input-adi-xml-prolog and xsl-stylesheet-prolog" on page 119. The following is an example of how to add a namespace definition for the item namespace to the xsl-stylesheet-prolog entry:

```
xsl-stylesheet-prolog = <?xml version='1.0' encoding='UTF-8'?>
<xml:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
xmll:items='http://mycompany/namespaces/items' version='1.0'>
  <xsl:output method='text' omit-xm1-declaration='yes'
    encoding='UTF-8' indent='no'/>
  <xsl:template match='text()'>
</xsl:template>
```

There are two ways to define a namespace prefix to the XML processor:

- Define the namespace globally for the entire XMLADI document.
- Define it individually within those ADI items that use the prefix.

In both cases, the namespace declaration must be included in the XML element’s start tag.

The first and simplest method of defining a namespace for the XML processor is to add the namespace definition to the XMLADI document element start tag. Adding the definition to the XMLADI document element start tag is easiest to do because it automatically defines the namespace for the entire document. Therefore, any ADI items in the document whose names are prefixed with this namespace ID do not have to have the namespace definition added to their own element start tag. This method does not suffer any of the drawbacks of defining the namespace by using the second method. The [xmladi-attribute-definitions] stanza has been added to the configuration file to allow customers to define namespaces globally for use within the XMLADI document. For information on how to add a namespace definition to the [xmladi-attribute-definitions] stanza, refer to "[xmladi-attribute-definitions]” on page 119.

The second method of specifying an XML namespace definition to the XML processor is to add the definition to the XML value of the ADI element itself. For example, given the ADI container name of item:name and a string value for the item of Widget A, to add the XML namespace definition to the XML item item:name, you would define item:name in XML as follows:
The ADI item:name must be added to an attribute list with an attribute name of item:name and the attribute's value would be the entire XML element definition above entered as a single contiguous text string. There are some drawbacks to defining the XML namespace within the XML definition of each ADI item rather than defining it globally for the entire XMLADI document. For instance, the value of any ADI items that use a namespace ID prefix must be in XML because the namespace definition can only be added to the XML definition of the value of the item, as demonstrated for item:name above. As a result items of ADI with namespace prefixes cannot simply have the value 100. The value of the item must an XML fragment, such as the string <prefix:adi_name>100</prefix:adi_name>.

Also, any ADI source that can provide values for namespace prefixed ADI items would need to ensure that the appropriate namespace definitions for the item are added to each XML formatted value that it returns. When the service does not normally return XML formatted data and is not aware of namespace prefixes, it must be changed so that it does, which translates to increased processing overhead for dynamic ADI retrieval services. By defining the namespace globally, all of these complications can be avoided. If a namespace has not been defined for either the XML or XSL processors, an error will be logged to the application error logs to the effect that the namespace ID does not have an associated URI mapping. This can happen during the creation of the rule if the XSL processor has not been notified of the new namespace, or during rule evaluation if the XML processor has not been notified.

### Authorization rules evaluator

The authorization rules evaluator evaluates authorization rules within the constraints required by the authorization engine.

The authorization rules evaluator takes the rule policy that is attached to the target protected object and evaluates the rule by calling the XSL processor. An input XML document for the transformation is constructed from the entitlements data defined in the credential of the user that is requesting the authorization, the application context passed in with the access decision call, the ADI required from the authorization engine, and the ADI retrieved from calls to dynamic ADI retrieval entitlement services.

The authorization engine expects the rules evaluation to result in the return of one of the string identifiers as shown in Table 3. The identifiers were chosen to ensure uniqueness in the event that the XSL rule is written incorrectly and the evaluation returns incorrect information. Delimiting the identifiers with an exclamation point (!) enables the evaluator to identify errant cases.

**Table 3. String identifiers returned by rules evaluation**

<table>
<thead>
<tr>
<th>Delimiter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>!TRUE!</td>
<td>Access is permitted.</td>
</tr>
<tr>
<td>!FALSE!</td>
<td>Access is denied.</td>
</tr>
<tr>
<td>!INDIFFERENT!</td>
<td>The rules engine has no opinion.</td>
</tr>
</tbody>
</table>

The identifiers should be the only text in the output document; although they can be surrounded by white space. If a value (other than the defined valid values or an
empty document) is returned, the access decision will fail and an error code will be returned to the resource manager to indicate that the rule is not compliant. The format of an authorization rule is outlined in "Format and constraints of rules."

In addition, the maximum length of any result text returned by a rule evaluation is limited to 1023 characters. Rules returning more text output than this causes the access decision to fail at runtime with a minor error code of ivacl_s_rule_result_string-too-large.

Format and constraints of rules

This section discusses the constraints that are placed upon XSL rules by the ADI XML model. This section does not go into detail explaining the XSL programming language, its syntax or set of standard supplied functions. The breadth of XSL processor functionality and capabilities of the language make documenting the language far too great a task for this document. There are a number of XSL (XSLT) programmer's guides available in bookstores that are better suited to this task.

An authorization rule must be defined as an XSL template within an XSL style sheet using the style sheet prolog specified in the configuration file. The rule must be written in a valid XSL template rule format and must return a text output document that contains one of the following string identifiers: TRUE!, FALSE!, or INDIFFERENT!. For more information on string identifiers, see "Authorization rules evaluator" on page 110.

For authorization decisions, the rule must return the expected decision data to the rules evaluator. The data that is returned from the rules-driven entitlements interface must be able to be expressed as a text name/value attribute pair in the entitlements output parameter of the azn_entitlement_get_entitlements() call. Many data providers return entitlements data in XML format; thus, no additional transformation is required to pass these entitlements into the rules evaluator as ADI.

All ADI that is passed to the rules evaluator must be specified in XML, and ADI that is passed in to the access decision or retrieved from the credential that is not in XML is formatted as such by the evaluator before an authorization rule can be evaluated.

The result of the XSL transformation performed by an XSL authorization rule must be a text output document containing only one of the string identifiers that are listed in Table 3 on page 110. The identifiers must be the only text in the output document but they can be surrounded by white space. The identifiers are not case sensitive. If a value other than one of those listed or an empty document is returned, the access decision will be failed by the authorization engine and an error code will be returned to the resource manager indicating that the rule is not compliant.

The following example references the XML data item defined in JohnSmith. The condition that the following example rule evaluates is expressed, as follows:

if ((AmountReqd + CreditCardBalance) < CreditCardLimit && MileagePlusStatus is "100k")

The corresponding authorization rule is:
<xsl:if test="(AmountReqd + JohnSmith/CreditCard/Balance)
    &lt; JohnSmith/CreditCard/Limit
    and JohnSmith/MileagePlus/MemberStatus = '100k'">
  !TRUE!
</xsl:if>

This example rule is the simplest form for specifying an authorization rule. It does not include its own template match statement and it accepts the default template match statement, which is set to /XMLADI. Template match statements are an XSL language construct that is used to select the point in the hierarchy of an XML document at which the XSL rules, which are contained within the template match statement, will be applied. The default template match statement of the ADI XML model matches from the top of the XMLADI document by specifying the XPath /XMLADI.

To add your own template match statement to a rule definition, only two additional lines are needed. For example, to rewrite the example to include your own explicit template match statement that matches from the root of the XMLADI document, you would modify the rule as follows:

<xsl:template match="/XMLADI">
  <xsl:if test="(AmountReqd + JohnSmith/CreditCard/Balance)
    &lt; JohnSmith/CreditCard/Limit
    and JohnSmith/MileagePlus/MemberStatus = '100k'">
  !TRUE!
</xsl:if>
</xsl:template>

To reference any data item in the document, the XPath to each node must include the XMLADI node. For example, to access the credit card balance, the full path would be /XMLADI/JohnSmith/CreditCard/Balance. When a rule is built, the rule writer must understand what the correct XPath, used to access the XML data nodes and subnodes, is from the current point in the tree. The current point in the tree is selected by using the template match statement. The template match statement allows an XSL programmer to shorten the XPath to each data element by specifying that the XPath processing occur further down the XML document tree.

The <xsl:template match="/XMLADI"> statement tells the XSL processor that all relative XPaths within the bounds of the template statement should be assumed to be relative to the node XMLADI. To shorten the XPaths even further, the template match statement could be set at /XMLADI/JohnSmith in which case, the credit card balance could be referred to as CreditCard/Balance.

Policy administrators must also make the following assumptions about the XSL style sheet document that is created by the rules evaluator to contain the rule that they devise:

- If a style sheet prolog is specified in the azn client configuration file, that prolog is imported into the empty style sheet. If no prolog is specified, the following default prolog is used instead:
  
  <!-- Required for XSLT language -->
  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      version="1.0">

  <!-- Required to constrain output of rule evaluation -->
  <xsl:output method="text" omit-xml-declaration="yes"
      encoding='UTF-8' indent="no"/>

  <!-- Need this to ensure default text node printing is off -->
  <xsl:template match="text()"></xsl:template>

112 IBM Tivoli Access Manager: Base Administration Guide
• Among other things, this prolog sets the XSL style sheet syntax to version 1.0, which is supported by the embedded XSL processor. The prolog sets the namespace for XSL documents to xs1, which requires that all XSL language-specific identities be prefixed by xs1:. This prefix is the standard mode of operation for XSL style sheets. Most attributes in this prolog must be in the style sheet or, if not, the results that are returned from the rules evaluator will not conform to the expected results.

• All authorization rules must be enclosed in an xsl:template match statement. If the rule is defined with its own xsl:template match statement, the rule will be accepted as is. This acceptance allows the rule creator to specify the level within the ADI XML document at which the rule matches data items. But in this case, the match statement must be the first statement encountered by the evaluator when validating the rule or it is assumed that there is no template match statement. If there is a match statement but the match statement does not begin with the absolute path /XMLADI, the rule will be returned as invalid. Relative match statements are not accepted at this level.

• If no match statement is specified in the rule, the rule is automatically enclosed in the following match statement:

```xml
<xsl:template match="/XMLADI">

<xsl:template>

```

• Therefore, all rules devised without an explicit template match statement must use XPath expressions that assume the XML context node is /XMLADI. The XPath expression for any ADI item must begin with the container name of the item and must be fully qualified.

Example rules
Following are example rules that demonstrate how rules can be implemented.

• “Example 1”

• “Example 2” on page 114

• “Example 3” on page 115

Example 1
This example is a simple rule that relies mostly on ADI passed in to the access decision call but it also requires an ADI container called printQuota to be stored in the requesting user credential or passed in as application context. The access decision logic defined by this rule is to permit access if this user is in the printUsers group, has requested a print operation (p) or has requested to queue a print job for printing later (q), and the quota of submitted print jobs for the day is less than 20.

```xml
<xsl:if test="azn_cred_groups = "cn=printUsers,o=ibm,c=us"
  and (contains(azn_engine_requested_actions,"p")
  or contains(azn_engine_requested_actions,"q"))
  and printQuota &lt;20">
  TRUE!
</xsl:if>
```

All other requests are denied. Note that the test condition for the group name returns an appropriate result regardless of the number of groups that the requesting user is in. The condition is an XSL node test that compares each value within the XML element azn_cred_groups with the DN string. It is important to note that the syntax for determining the opposite case (for example, that the requesting user is not in the printUsers group) requires a slightly different expression because it is a node test. Refer to “Example 2” on page 114 for an
example of how to test for whether a set of values like the `azn_cred_group_names` attribute does not contain a certain member.

**Example 2**
In the following example, the rule works on data that is within the authorization credential. It evaluates the following attributes:

- `azn_cred_principal_name`
- `azn_cred_groups`
- `azn_cred_registry_id`

Each of the `xsl:when` statements is evaluated. The first statement with conditions that are all true returns a result. Each condition tested has a comment that explains its action.

```xml
<!-- Example choose rule -->
<xsl:choose>
  <!-- Explicitly allow if the requesting user is myuser0 -->
  <xsl:when test="azn_cred_principal_name = 'myuser0'">
    !TRUE!
  </xsl:when>

  <!-- Explicitly deny if the requesting user is myuser1 -->
  <xsl:when test="azn_cred_principal_name = 'myuser1'">
    !FALSE!
  </xsl:when>

  <!-- Explicitly allow if the requesting user's LDAP DN is the same as that specified -->
  <xsl:when test="azn_cred_registry_id = 'cn=myuser3,secAuthority=Default'">
    !TRUE!
  </xsl:when>

  <!-- This rule permits access to any user who is a member of mygroup1 but is not a member of mygroup2 -->
  <xsl:when test="azn_cred_groups = 'mygroup1' and not (azn_cred_groups = 'mygroup2')">
    !TRUE!
  </xsl:when>

  <xsl:otherwise>
    !FALSE!
  </xsl:otherwise>
</xsl:choose>
```

The fourth `xsl:when` statement uses the `not()` function to negate the Boolean result of the following test:

`azn_cred_groups = 'mygroup2'`

The `not()` function is used instead of the valid authorization rule operator `!=` operator because, in this case, the `azn_cred_groups` attribute is a multi-valued attribute. Multi-valued attributes like `azn_cred_groups` return a set of values, referred to as a `node-set` in XSL, to be tested by the condition. Each node value in the set is tested against the condition individually and `!TRUE` is returned if any of the conditions evaluate to true. In any case, where the user is in more than one group, other than `mygroup2`, the result of the node test is always `!TRUE`. To test the nonexistence of something in a node-set, use the `not()` function instead of the `!=` operator. For example, you can test that the condition `group is mygroup2` is not true.
**Example 3**

This example evaluates application-defined XML data. The XML object that it expects is a batch object that contains a list of operations that are to be performed together. The batch object consists of any number of transaction elements, which consist of an item and the amount of those items to order.

<!-- batched transaction -->

```xml
<batch>
  <max_tx_count>5</max_tx_count>
  <max_tx_amount>150</max_tx_amount>
  <account>customerA</account>
  <transaction>
    <item>widgetA</item>
    <amount>10</amount>
  </transaction>
  <transaction>
    <item>widgetB</item>
    <amount>20</amount>
  </transaction>
  <transaction>
    <item>widgetC</item>
    <amount>30</amount>
  </transaction>
  <transaction>
    <item>widgetD</item>
    <amount>40</amount>
  </transaction>
  <transaction>
    <item>widgetE</item>
    <amount>50</amount>
  </transaction>
</batch>
```

The authorization rule checks that the requesting user is a member of a group whose name matches the name of the account in the transaction (in this example, it is customerA). If the requesting user is not a member of this group, the user will not be authorized to submit batch requests on behalf of customerA. Then, the rule checks that the total number of transactions within the batch is less or equal to the `max_tx_count` element of the batch object and that the total number of items ordered in the entire request is less than the `max_tx_amount` element of the batch object. The rule calls the `<count()>` and `<sum()>` functions. The `<count()>` function counts the number of instances of a transaction element within the batch. The `<sum()>` function totals the value of all of the amount elements within all transaction elements in the batch. The rule is as follows:

<!-- Compare group to batch customer and num transactions and total tx amounts to limits. -->

```xml
<xsl:if test="azn_cred_groups = batch/account 
  and count (batch/transaction) &lt;= batch/max_tx_count 
  and sum (batch/transaction/amount) &lt;= batch/max_tx_amount">
  !TRUE!
</xsl:if>
```
Methods of providing ADI to the rules evaluator

A resource manager application can provide ADI from the resource manager to the rules evaluator in one of two ways:

- Adding the attributes to the application context parameter
- Configuring to supply the missing ADI to the authorization engine only when it is explicitly requested

The first method is to provide the ADI by adding the attributes to the application context parameter passed into the `azn_decision_access_allowed_ext()` call. The problem with this method is that the resource manager must either know which ADI is going to be needed by any particular access decision up front or, more likely, provide all of the ADI for all known rules to the authorization engine for every access decision call regardless of whether a rule is involved in the decision.

The first method might be acceptable and even desirable for a smaller set of ADI. However, for a larger and more varied set of possible ADI, a second method is needed. You can configure the resource manager to supply the missing ADI to the authorization engine only when it is explicitly requested. With this method, the authorization engine can be configured with a set of ADI prefixes that can be provided by the resource manager upon request. The authorization engine fails the access decision and notifies the resource manager of the ADI it needs in a permission information attribute returned by the `azn_decision_access_allowed_ext()` call. The attribute contains a list of the ADI that is needed to successfully evaluate the rule. The ADI was not found in the application context that was passed in and did not have a prefix matching those that the resource manager has identified as its own.

The permission information attribute is named `azn_perminfo_rules_adi_request` and contains a text attribute value for each item of ADI required. The resource manager looks for this attribute when the access decision fails, and in the event that it is present, scans the list of ADI names in the attribute and gathers the requested data to retry the access decision with this additional data. If the requested data cannot be provided, the resource manager should deny access and log the problem as a failure due to insufficient rules data. The requested list contains only the ADI items that are identified as being provided by the resource manager. The unique prefix added to the attribute name is used to identify the ADI. All resource managers that provide data to the evaluation process in this manner must define a unique prefix by which their ADI data set can be identified.

Permission information is returned to a resource manager application only when the authorization client has been configured that way. To activate the return of the `azn_perminfo_rules_adi_request` permission information attribute, the name of this attribute must either be added to the `azn_init_set_perminfo_attrs` initialization attribute or the equivalent configuration file stanza entry `permission-info returned in the [aznapi-configuration] stanza`

The ADI prefixes that are recognized by the resource manager can be configured using the `resource-manager-provided-adi` configuration file entry or the `azn_init_resource_mgr_provided_adi` initialization attribute. See page 251 for the configuration file stanza entry, or see the “resource-manager-provided-adi” on page 118 for more explanation. For more a discussion about the `azn_init_resource_mgr_provided_adi` initialization attribute, refer to the IBM Tivoli Access Manager for e-business Authorization C API Developer Reference.
The authorization engine attempts to anticipate the need to request information from the resource manager by obtaining the rule policy object on the protected object early in the access decision process. The authorization engine then compares the required ADI in the rule with the ADI names in the application context parameter that is passed by the resource manager. The ADI names, which are missing from the application context and which are specific to the resource manager, are added to the returned permission information object.

The ADI prefixes must be unique to identify them as the resource manager ADI and to avoid conflict with ADI provided within the credential, from the authorization engine and by other external data providers.

**Reason codes for rule failures**

When this mode of operation is selected, the authorization engine processes all policies for the access decision as normal but if the rule evaluation fails, the engine returns access denied along with a reason code for the rule failure in the permission information attribute list. The permission information attribute name is `azn_perminfo_reason_rule_failed`.

This feature allows the target application to fail or permit the access request based on the rule failure reason code it is given by the resource manager. When access is denied, the application must check the permission_info attribute list returned from the access decision call to determine if a rule failure reason code was returned from the access decision. The resource manager does not need to check for the attribute on a successful access decision call. The IBM Tivoli Access Manager for e-business application is an example of an aznAPI resource manager that can make use of the rule failure reason code. When configured to do so, IBM Tivoli Access Manager for e-business forwards the rule failure reason code on to the protected Web application in an HTML environment variable. The protected Web application must be mounted through a secure junction to have access to the rule failure reason code defined for the authorization rule. The use of rule failure reason codes in IBM Tivoli Access Manager for e-business is limited to the protected object space of junctioned Web applications.

The attribute value (the reason code) of the `azn_perminfo_reason_rule_failed` attribute is a single string whose value is determined and defined by the policy administrator and is set in the rule policy object when it is first created. The only constraint on the value of the reason code is that the value must be a string.

The following conditions must be met before a rule failure reason code is returned to the caller:

- The reason code is returned only when the access request is denied and the rule policy evaluation denies access, but not for every case in which access is denied. The reason code is not returned when the rule evaluation succeeds. The rule failure reason code will not be returned if the rule failed due to a rule syntax error or if there was insufficient ADI to perform the rule evaluation. In these latter cases, the authorization decision is failed with an error status.
- There must be a reason code set in the attached rule policy object. This value is set in the rule policy using the admin API or the `pdadmin` commands.
- The aznAPI application must be enabled to return the rule failure reason as permission information. To do this, either the `azn_init_set_perminfo_dims` initialization parameter or the equivalent configuration file entry in the `[aznapi-configuration]` stanza (stanza entry `permission-info-returned`) must include the attribute name `azn_perminfo_reason_rule_failed`. This enables the attribute to be returned by the authorization engine in the permission

Chapter 10. Authorization rules management 117
information output parameter (perminfo) of azn_decision_access_allowed_ext(). For more information on permission information attributes and how to configure the authorization engine to return them refer to the IBM Tivoli Access Manager for e-business Authorization C API Developer Reference.

Configuration file and initialization attributes

A number of configuration file entries and initialization attributes have been defined to control aspects of the initialization of the rules evaluator within the authorization engine. The configuration entries are placed in the resource manager’s configuration file. An example of this aznAPI.conf configuration file is provided in the example/authzn_demo/cpp directory of the Tivoli Access Manager Application Developer Kit (ADK) package. Configuration files are also used by Tivoli Access Manager resource management applications, such as IBM Tivoli Access Manager for e-business, and these configuration entries can be added to the configuration file of these applications. Refer to the documentation for the specific Tivoli Access Manager application for more information on the applications configuration file.

Initialization attributes are the programmatic equivalent of configuration attributes and are intended to be used to develop a custom resource manager application. For more information on the authorization-rule-specific initialization attributes and the process of developing a custom resource manager aznAPI application, refer to the IBM Tivoli Access Manager for e-business Authorization C API Developer Reference.

resource-manager-provided-adi

The resource-manager-provided-adi configuration stanza entry defines the prefixes that the authorization engine uses to determine the set of missing ADI that is provided by the resource manager. This entry uses a string prefix as its value. To specify more than one prefix you must add multiple stanza entries as in the following examples:

```
resource-manager-provided-adi = sales_customer_
resource-manager-provided-adi = sales_item_
```

These examples notify the authorization engine that any ADI it requires that begins with sales_customer_ or sales_item_ be provided by the resource manager application. ADI items named sales_customer_name, sales_customer_address, sales_item_count, and sales_item_price are examples of ADI that the authorization engine would request from the resource manager.

dynamic-adi-entitlement-services

The dynamic-adi-entitlement-services configuration entry lists the service IDs of the dynamic ADI retrieval entitlement services that should be called by the authorization engine in the event that ADI is found to be missing from the requesting user’s credential or from the application context, and cannot be gathered from the resource manager. Any entitlement service configured under this entry is called by the authorization engine using the azn_entitlement_get_entitlements() interface and is passed the azn_perminfo_rules_adi_request attribute. The string values of this attribute are the container names of the ADI that are still required. If the dynamic ADI retrieval service can fulfill the request, it returns the requested data to the authorization engine in the entitlements parameter. Examples of entitlement services that can be used in this manner are the Cred Attributes Entitlement Service and the Entitlement Service Demo, both of which are provided with Tivoli Access Manager.
For more information on how to configure and use these entitlement services refer to the IBM Tivoli Access Manager for e-business Authorization C API Developer Reference.

To specify that the authorization engine should call multiple dynamic ADI retrieval services, you must specify multiple entries. The following examples demonstrate how to specify the service IDs of two different entitlement services for use as dynamic ADI entitlement services. The service IDs must correspond to valid entitlement service definitions in the [aznapi-entitlement-service] stanza.

dynamic-adi-entitlement-services = ent_cred_attrs_id

dynamic-adi-entitlement-services = ent_svc_demo_id

**input-adi-xml-prolog and xsl-stylesheet-prolog**

The input-adi-xml-prolog and xsl-stylesheet-prolog configuration entries have been defined to allow augmentation of the XML and XSL prolog statements that are prepended to the ADI XML document and authorization rule stylesheet before they are passed to the rules evaluator for processing. The format and defaults for each of these entries are:

```xml
input-adi-xml-prolog=<?xml version="1.0" encoding="UTF-8"?>
```

and

```xml
xsl-stylesheet-prolog=<?xml version="1.0" encoding='UTF-8'?>
<xsl:stylesheet xmlns:xsl='http://www.w3.org/1999/XSL/Transform' version='1.0'>
<xsl:output method = 'text' omit-xml-declaration='yes' encoding='UTF-8' indent='no'/>
<xsl:template match='text()'>
</xsl:template>
</xsl:stylesheet>
```

Due to the constraints imposed by the authorization rule model, there are a number of prolog attributes that are required by the authorization engine (all of which are specified in the default prolog entries). If any of these attributes are changed or omitted from the entry, the authorization client fails to start and returns an error.

**Note:** Be sure that you are familiar with the Xalan XSL processor, the Xerces XML processor, and the use of prolog statements before any attempt is made to change these entries from the defaults provided.

**[xmladi-attribute-definitions]**

The [xmladi-attribute-definitions] stanza enables customers to add XML attribute definitions, such as XML namespace definitions, to the XMLADI document start tag. For example, when an application wants to use namespaces to differentiate or aggregate ADI items, as discussed in "Defining an XML namespace" on page 108, the XML processor must be notified of the namespace by using an XML namespace definition. The namespace definition can be added to this stanza, and it is automatically added to the XMLADI document element start tag. The benefit of adding definitions such as these to the XMLADI document start tag using this mechanism is that the attribute definitions are then available for all ADI items defined in the XMLADI document, whether their values were retrieved from the credential, generated by the authorization engine or retrieved by a dynamic ADI entitlement service. For example:

```xml
<xml:myNS = "http://myURI.mycompany.com"
appID = "Jupiter" - Account Management Web Portal Server #1.'
```
Managing an authorization rule

The following tasks related to authorization rules can be performed:

- “Create an authorization rule”
- “List authorization rules” on page 121
- “Delete an authorization rule” on page 121
- “Modify an authorization rule” on page 122
- “Attach an authorization rule to a protected object” on page 122
- “Finding objects having authorization rules attached” on page 123
- “Detaching an authorization rule” on page 124

Note: When entering rule text on the pdadmin command line, it is necessary to enclose the rule text within double quotation marks ("). As a result, any double quotation marks embedded within the rule text must be escaped with a backslash so that they are ignored by the pdadmin command line processor. The XSL processor treats single and double quotation marks equally for the purpose of defining text strings so they can be used interchangeably, but must always be paired appropriately.

For example:

```
padmin sec_master> authzrule create testrule1
   "<xsl:if test='some_piece_of_ABI ="any_string"'>!TRUE!</xsl:if>
```

Create an authorization rule

To create an authorization rule using the Web Portal Manager or the pdadmin command line utility, log in to the desired domain as a domain administrator.

Web Portal Manager

To create a new rule using Web Portal Manager, do the following:

1. Log in to the domain.
2. Click AuthzRule → Create AuthzRule to display the Create AuthzRule window.
3. Type the AuthzRule Name for the authorization rule that you want to create (for example, r2). This field is required.

   Note: It is recommended that you not use the following characters in the name of a rule:
   
   | ! | @ | # | $ | & | ( | ) | * | + | , | ; | : | < | > | = | @ | 

4. Type a description of the authorization rule in the Description field. For example:
   
   time-of-day rule for engineering object space

5. Type the text of the rule policy in the AuthzRule Text field. This field is required. For example, you might copy the text that is contained in a file named engineering.xsl into this field.

6. Type the text that you want to be returned to the resource manager if the rule denies access to a protected object in the Fail Reason field (for example, error.
7. Click **Create**. If successful, the new rule is displayed as a link on the Manage AuthzRules window. If you select the authorization rule link, the properties of that rule will be displayed.

**pdadmin**
To create an authorization rule using the **pdadmin** utility, log in to the domain and use the **authzrule create** command. For example, to create a rule named r2 with a rule file named engineering.xsl that implements the time-of-day rule for the engineering object space and returns a fail reason code of error, enter the following:

```
pdadmin sec_master> authzrule create r2 -rulefile engineering.xsl
   -desc "time-of-day rule for engineering object space"
   -failreason error
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference***.

**List authorization rules**
To list the authorization rules that have been created using the Web Portal Manager or the **pdadmin** command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**
To list all existing authorization rules using Web Portal Manager, do the following:
1. Log in to the domain.
2. Click **AuthzRule > List AuthzRule** to display the Manage AuthzRules window. A list of names for authorization rules that have been created in Tivoli Access Manager are displayed as links. If you select an authorization rule link, the properties of that rule will be displayed.

**pdadmin**
To list authorization rules in the domain using the **pdadmin** utility, log in to the domain and use the **authzrule list** command.

```
pdadmin sec_master> authzrule list
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference***.

**Delete an authorization rule**
To delete an authorization rule using the Web Portal Manager or the **pdadmin** command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**
To delete an authorization rule using Web Portal Manager, do the following:
1. Log in to the domain.
2. Click **AuthzRule > List AuthzRule** to display the Manage AuthzRules window. A list of authorization rules that have been created in Tivoli Access Manager are displayed. Each rule is a link that displays properties for that rule when selected.
3. Select one or more check boxes for the links that you want to delete. For example, you might select the check box for the authorization rule named r2.
4. Click **Delete** to display the Delete AuthzRules window where you are prompted to confirm or cancel the deletion.
To delete an authorization rule in the domain using the `pdadmin` utility, log in to the domain and use the `authzrule delete` command. For example, to delete a rule named `r2`, enter the following:

```
pdadmin sec_master> authzrule delete r2
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*.

**Modify an authorization rule**

To modify an authorization rule using the Web Portal Manager or the `pdadmin` command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**

To modify an authorization rule using Web Portal Manager, do the following:

1. Log in to the domain.
2. Click **AuthzRule → List AuthzRule** to display the Manage AuthzRules window.
   A list of authorization rules that have been created in Tivoli Access Manager are displayed. Each rule is a link that displays properties for that rule when selected.
3. Click the authorization rule link for the rule that you want to change. The AuthzRule Properties–General window is displayed.
4. As needed, change the following information:
   - The description
   - The authorization rule text
   - The fail reason

   For example, if no description currently exists, add a description. If a description currently exists, change the authorization rule description by typing the new description in the **Description** field (such as add the words updated June 23 2003):

   updated June 23 2003 time-of-day rule for engineering object space

5. Click **Apply** for the changes to take effect.

To modify an authorization rule in the domain using the `pdadmin` utility, log in to the domain and use the `authzrule modify` command. For example, to change the rule named `r2` to return a fail reason code of warning, enter the following:

```
pdadmin sec_master> authzrule modify r2 failreason warning
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*.

**Attach an authorization rule to a protected object**

To attach an authorization rule to a protected object using the Web Portal Manager or the `pdadmin` command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**

To attach a rule to a protected object using Web Portal Manager, do the following:

1. Log in to the domain.
2. Click **AuthzRule → List AuthzRule** to display the Manage AuthzRules window.
A list of authorization rules that have been created in Tivoli Access Manager are displayed. Each rule is a link that displays properties for that rule when selected.

3. Click the link for the authorization rule that you want to attach to an object (for example, the r2 authorization rule). The AuthzRule Properties–General window is displayed.

4. Click the **Attach** tab to view a list of protected objects to which the authorization rule is already attached, if any.

5. Click **Attach** to display the Attach AuthzRule window.

6. Type the **Protected Object Path** of the protected object to which you want to attach the authorization rule. This field is required. Be sure to type the full path name. For example:

   /WebSEAL/tivoli.com/w3junction/index.html

7. Click **Attach**. If successful, the new protected object will be added as a link to the list of objects to which the authorization rule is attached on the AuthzRule Properties–Attach window.

**pdadmin**

To attach an authorization rule to a protected object using the **pdadmin** utility, log in to the domain and use the **authzrule attach** command. For example, to attach a rule named r2 to a protected object named /WebSEAL/tivoli.com/w3junction/index.html, enter the following:

```
pdadmin sec_master> authzrule attach /WebSEAL/tivoli.com/w3junction/index.html r2
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference.*

**Finding objects having authorization rules attached**

To find the protected objects that an authorization rule is attached to using the Web Portal Manager or the **pdadmin** command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**

To find the protected objects that are attached to a rule using Web Portal Manager, do the following:

1. Log in to the domain.
2. Click **AuthzRule → List AuthzRule**. A list of authorization names is displayed. Each authorization rule name is a link that you can click to display the AuthzRule Properties window.
3. Click the **Attach** tab.

**pdadmin**

To find all the protected objects to which an authorization rule is attached in the domain by using the **pdadmin** utility, log in to the domain and use the **authzrule find** command. For example, to find the protected objects attached to a rule named r2, enter the following:

```
pdadmin sec_master> authzrule find r2
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference.*
Detaching an authorization rule

To detach an authorization rule from a protected object using the Web Portal Manager or the `pdadmin` command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**

To detach a rule from a protected object using Web Portal Manager, do the following:

1. Log in to the domain.
2. Click `AuthzRule` → `List AuthzRule` to display the Manage AuthzRules window. A list of authorization rules that have been created in Tivoli Access Manager are displayed. Each rule is a link that displays properties for that rule when selected.
3. Click the link for the authorization rule that you want to detach from an object (for example, the r2 authorization rule). The AuthzRule Properties–General window is displayed.
4. Click the `Attach` tab to view a list of protected objects to which the authorization rule is already attached, if any.
5. Select one or more check boxes for the protected objects from which you want to detach the authorization rule.
6. Click `Detach` to display the Detach AuthzRule from Object window where you are prompted to confirm or cancel the request.

**pdadmin**

To detach a rule from a protected object in the domain using the `pdadmin` utility, log in to the domain and use the `authzrule detach` command. For example, to detach a rule from a protected object named `/WebSEAL/tivoli.com/w3junction/index.html`, enter the following:

```
pdadmin sec_master> authzrule detach /WebSEAL/tivoli.com/w3junction/index.html
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*. 
Chapter 11. User and group management

An initial domain administrator is created when a new domain is created. The domain administrator has the necessary privileges to manage the domain at the outset. The domain administrator can be used to create and configure users, groups, resources, and applications, and to delegate administration tasks within the domain as required.

A user represents any authenticated Tivoli Access Manager identity. Typically, users represent network users or resource managers.

A group is a collection of one or more users. An administrator can use group ACL entries to easily assign the same permissions to multiple users. New users to the domain gain access to objects by becoming members of appropriate groups. Group membership eliminates the need to create new ACL entries for every new user. Groups can represent organizational divisions or departments within a domain. Groups are also useful in defining roles or functional associations.

Accounts refer to users and groups collectively.

A registry unique identifier (Registry UID) specifies the location in the registry database where the new user is created. Similarly, a registry group unique identifier (Registry GID) specifies the location in the registry database where the new group is created. For registry UIDs and GIDs, you must type the full path name for the new user or group. The path format depends on the type of registry that the product is using:

- LDAP: `cn=IBM-Support,o=ibm,c=us`
- Active Directory: `cn=IBM-Support,dc=Austin,dc=US`
- Domino: `IBM-Support/Austin/US`

The Registry UID or GID provides extra security in the case where a user or group is deleted from the domain and then recreated with the same name. For example, even though a new user has the same name as the deleted user, Tivoli Access Manager allocates a new Registry UID to this user. Because the Registry UID is new, any existing ACL entries that refer to the old user name do not grant any rights to the new user. Stale UIDs from deleted users and groups are silently removed by the policy server (pdmgrd).

Search for users

To search for users using the Web Portal Manager or the padmin command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**

To search for and list up to a maximum of 100 users using Web Portal Manager:

1. Log in to the domain.
2. Click Search Users.
3. At the User Search page, use the wildcard * to filter user ID names. This field is required.
4. Use the default value of 100 or type a Maximum Results number to limit the number of user IDs that you want to view.
5. Click **Search** to display a table of user IDs. Each user ID is displayed as a link. From the User Search window, you can perform these tasks: create a new user, delete one or more existing users, and click on the link to view user properties.

6. Use the default value of 15 user IDs per page. Or click **Options** to enter the number of user IDs you want to view per page; toggle back by clicking **Hide Options**.

7. Use the default value of None, meaning no text is used for filtering. Or, click **Filters** to find user IDs that contain, start with, or end with the text that you specify; toggle back by clicking **Hide Filters**.

**pdadmin**

To search for a list of users using **pdadmin** command line utility, log in to the desired domain as a domain administrator.

For example, to search for and list up to a maximum of 100 users, enter:

```
pdadmin sec_master> user list * 100
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*.

---

**Create a user**

To create a user using the Web Portal Manager or the **pdadmin** command line utility, log in to the desired domain as a domain administrator.

**Note:** When a user is created, the domain administrator assigns a user name (sometimes also referred to as a principal name). The user name must be unique within the domain because it is used by Tivoli Access Manager to identify this user. A registry user identifier, known as a distinguished name (DN), is also assigned to uniquely identify the user definition in the user registry. The format of the DN depends on the registry type being used. Also assigned are the common name (CN) and surname (SN) of the user being defined.

**Web Portal Manager**

To create a user to the domain using Web Portal Manager:

1. Log in to the domain.
2. Click **User → Create User**.
3. Type a **User Id** (for example maryj). This field is required.
4. Click **Group Membership** to search for groups in which the user can be a member.
5. Type the **First Name** (given name) of the user (for example Mary). This field is required.
6. Type the **Last Name** (family or surname) of the user (for example Jones). This field is required.
7. Type a **Password**. This field is required. Passwords must adhere to the password policies that are set by the domain administrator.
8. Type the password again to **Confirm Password**. This field is required.
9. Type a **Description** for the user ID (for example, Member of Marketing Group).
10. Type a **Registry UID**. This field is required. The registry UID specifies the location in the registry database where the new user is created. For example:
Import a Web pdadmin user Portal

5. Access to Reference

For that Remember

“Mary” pdadmin the utility, To create an account ability is informed the signon) administrator.

Select the Account Valid check box to indicate that the new user has the ability to participate in the domain. If this option is not selected, the new user account will not be valid and that user will be unable to log in.

12. Select GSO User check box to indicate the use of Global Sign-on (single signon) capability of Tivoli Access Manager.

13. Select the Password Valid check box to force a password change the next time the user logs in to the domain. If this option is not selected, the user will be informed that the password has expired.

14. Click No Password Policy to indicate that you do not want the initial password to conform to the password policies that are set by the domain administrator.

15. Click Create. A message will be shown if the user ID is created.

\textbf{pdadmin}

To create a user using the Web Portal Manager or the pdadmin command line utility, log in to the desired domain as a domain administrator.

For example, to create the user named maryj with Global Sign-on capability, enter the following pdadmin command on one line:

```
pdadmin sec_master> user create –gsouser maryj "cn=Mary Jones,o=IBM,c=us,dc=mkt"
"Mary" "Jones" pwd2pwd2
```

Remember that the format of the distinguished name depends on the registry type that is being used.

For more information, see the \textit{IBM Tivoli Access Manager for e-business Command Reference}.

\section*{Import a user}

To import a user that already exists in a user registry and make that user a Tivoli Access Manager user using the Web Portal Manager or the pdadmin command line utility, log in to the desired domain as a domain administrator.

\textbf{Note}: When a user is imported, the domain administrator assigns a user name (sometimes also referred to as a principal name). The user name must be unique within the domain because it is used by Tivoli Access Manager to identify this user.

\section*{Web Portal Manager}

To import a user that already exists in a user registry and make that user a Tivoli Access Manager user using the Web Portal Manager:

1. Log in to the domain.
2. Click User \textgreater Import User.
3. Type a User Id (for example maryj). This field is required.
4. Click Group Membership to search for groups in which the user can be a member.
5. Type a Registry UID. This field is required. The registry UID specifies the location in the registry database to be imported. For example:
cn=maryj,o=ibm,c=us,dc=mkt. Lotus Notes users require the full path name for the user being imported. For example: Mary Jones/IBM/US

6. Select the **Account Valid** check box to indicate that the new user has the ability to participate in the domain. If this option is not selected, the new user account will not be valid and that user will be unable to log in.

7. Select **GSO User** check box to indicate that this user can use the Global Sign-on capability of Tivoli Access Manager.

8. Select the **Password Valid** check box to force a password change the next time the user logs in to the domain. If this option is not selected, the user will be informed that the password has expired.

9. Click **Create**. A message will be shown if the user ID is created.

**pdadmin**

To import a user that already exists in a user registry and make that user a Tivoli Access Manager user using the **pdadmin** command line utility, log in to the desired domain as a domain administrator.

For example, to import the user information for the user named **maryj** from the existing user registry definition, enter:

```
pdadmin sec_master> user import -gsouser maryj "cn=Mary Jones,o=IBM,c=us,dc=mkt"
```

**Note**: The user information that is imported to the domain can be imported again to another domain.

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference.*

**Set global user policies**

You can change global user settings, such as password policies, login-failure policies, access policies, and account expiration policies.

**Note**: The valid range for numbers can be any number. However, a reasonable number should be used for the task you are wanting to perform. For example, a minimum password length should be long enough to protect your system but not so short as to make it easy for someone to determine your password by trying different combinations.

**Web Portal Manager**

To change global user settings using Web Portal Manager:

1. Log in to the domain.
2. Click **User → Show Global User Policy**.
3. Select **Unset** or **Set** for **Max Login Failures** to set or unset the maximum number of login failures before the account is no longer allowed to participate in the secure domain. If you select **Set**, either accept the default value of 10 or change the value to a number equal to or greater than zero.
4. Select **Unset**, **Disable**, or **Set** for **Disable Time Interval** to set the time, in seconds, or to disable each user account when the maximum number of login failures is exceeded. If you select **Set**, either accept the default value of 180 seconds or change the value to a number equal to or greater than zero.
5. Select Unset or Set for **Minimum Password Length** to set the minimum number of characters required for the password. If you select Set, either accept the default value of 8 alphanumeric characters or change the value to a number greater than zero.

6. Select Unset or Set for **Maximum Password Age** to set the maximum time a password can be used before it expires. The maximum password age is relative to the last time the password was changed. If you select Set, either accept the default value of 91 days (91:00:00:00) or change the value to a number greater than zero.

7. Select Unset or Set for **Minimum Password Alphas** to set the minimum number of alphabetic characters required in a password. If you select Set, either accept the default value of four alphabetical characters or change the value to a number greater than one.

8. Select Unset or Set for **Minimum Password Non-Alphas** to set the minimum number of non-alphabetic characters required in a password. If you select Set, either accept the default value of one non-alphabetical character or change the value to a number greater than one.

9. Select Unset or Set for **Max Password Repeated Characters** to set the maximum number of repeated characters allowed in a password. If you select Set, either accept the default value of two repeated characters or change the value to a number greater than two.

10. Select Unset, Yes, or No for **Password Spaces Allowed** to determine whether spaces are allowed in passwords. You can accept the default setting of Unset. Or, you can change the value to Yes to allow spaces in passwords or to No to not allow spaces in passwords.

11. Select Unset, Unlimited, or Set for **Account Expiration Date** to set the account expiration date. You can accept the default setting of Unset. Or, you can change it to Unlimited or Set.

   If you select Set, type the 4-digits year in the **Year** field.

   Either accept the default value of Jan 01-00:00:00 or change the value to the date and time, specified as Month DD:hh:mm:ss. The hours must be entered using a 24-hour clock (for example, 09 for 9:00 a.m. or 14 for 2:00 p.m.).

12. Select Unset or Set for **Time of Day Access** to set the time of day access policy. If you select Set, either accept the default settings or change them. You can change these values:

   • Select the days of the week from the choices provided.

   • Select All Day or Between hours of.

   If you select Between hours of, also select the Start Time. The start time format is specified as hours and minutes. The start time is expressed by using a 24-hour clock.

   If you select Between hours of, also select the End Time. The end time format is specified as hours and minutes. The end time is expressed by using a 24-hour clock.

   If you select Between hours of, also select Local Time or UTC Time. The time zone is local by default; UTC is coordinated universal time.

13. Click **Apply** to put your changes into effect.

**pdadmin**

To set or change global user settings information using the pdadmin utility, log in to the domain as a domain administrator and use the **pdadmin policy set** commands.
For example, to set a global user policy to a maximum password age of 31 days 8 hours and 30 minutes, enter:

```
pdadmin sec_master> policy set max-password-age 031-08:30:00
```

For more information, see the IBM Tivoli Access Manager for e-business Command Reference.

**Change user password**

To change a user’s password using the Web Portal Manager or the `pdadmin` command line utility, log in to the desired domain as a domain administrator. The new password must comply with password policies that are currently in effect. See “Set global user policies” on page 128.

**Web Portal Manager**

To change the password for the specified user ID using Web Portal Manager:

1. Log in to the domain.
2. Click User → Change My Password.
3. Verify that the User ID identifies the login identifier for the user whose password you want to change.
4. Type the Current Password for the specified user ID. This field is required.
5. Type the New Password for the specified user ID. This field is required.
6. Type the password again in the Confirm New Password field. This field is required.
7. Click Apply to put the new password into effect.

**pdadmin**

To change the password for the user using the `pdadmin` utility, log in to the domain as a domain administrator and use the `pdadmin user modify user_name password` command.

For example, to change the password to `newpasswd` for user account `dlucas`, enter:

```
pdadmin sec_master> user modify dluca5 password newpasswd
```

For more information, see the IBM Tivoli Access Manager for e-business Command Reference.

**Search for groups**

To search for group names using the Web Portal Manager or the `pdadmin` command line utility, log in to the desired domain as a domain administrator.

**Web Portal Manager**

To search for and list up to a maximum of 100 groups using Web Portal Manager:

1. Log in to the domain.
2. Click Search Groups.
3. At the Group Search page, use the wildcard * to filter group names. This field is required.
4. Use the default value of 100 or type a Maximum Results number to limit the number of group names that you want to view.
5. Click **Search** to display a table of group names. Each group name is displayed as a link.

   From the Group Search window, you can perform these tasks: create a new group, delete one or more existing groups, and click on the link to view group properties.

6. Use the default value of 15 group names per page. Or click **Options** to enter the number of group names you want to view per page; toggle back by clicking **Hide Options**.

7. Use the default value of **None**, meaning no text is used for filtering. Or, click **Filters** to find group names that contain, start with, or end with the text that you specify; toggle back by clicking **Hide Filters**.

**pdadmin**

To search for a list of groups using **pdadmin** command line utility, log in to the desired domain as a domain administrator.

For example, to search for and list up to a maximum of 100 groups, enter:

```
padmin sec_master> group list * 100
```

For more information, see the *IBM Tivoli Access Manager for e-business Command Reference*.

---

### Create a group

To create a group using the Web Portal Manager or the **pdadmin** command line utility, log in to the desired domain as a domain administrator.

**Note:** When a group is created, the domain administrator assigns a group name. The group name must be unique within the domain because it is used by Tivoli Access Manager to identify this group.

For more information on groups, see [“Create groups” on page 162](#).

#### Web Portal Manager

To create a group in the domain using Web Portal Manager:

1. Log in to the domain.
2. Click **Group > Create Group**.
3. Type a **Group Name** for the group (for example, **sales**). This field is required.
4. Optionally, type a **Description** for the group (for example, **Sales**).
5. Type a **Registry GID**. This field is required. The registry GID specifies the location in the registry database where the new group is created. For example: `cn=Sales,o=ibm,c=us,dc=mkt`. Lotus Notes users require the full path name for the user being created. For example: `Sales/IBM/US`.
6. Optionally, type the path in the **Object Container field** to the Tivoli Access Manager object space where the group is to be created.
7. Click **Create**. The new group will be displayed as a link. Select the link and the properties for the new group are displayed.

**pdadmin**

To create a group in the domain using the **pdadmin** utility, log in to the domain as a domain administrator and use the **pdadmin group create** command to create a
new group and optionally place this group in a group container object. If the container object does not currently exist, it is automatically created. For example, to create the group named sales, enter:

```
$ pdadmin sec_master> group create sales "cn=sales,o=IBM,c=us,dc=mkt" Sales
```

For more information, see the IBM Tivoli Access Manager for e-business Command Reference.

---

**Import a group**

To import an existing group from a user registry into the domain and make that group a Tivoli Access Manager group using the Web Portal Manager or the `pdadmin` command line utility, log in to the desired domain as a domain administrator.

**Note:** When a group is imported, the domain administrator assigns a group name. The group name must be unique within the domain because it is used by Tivoli Access Manager to identify this group.

For more information on groups, see “Create groups” on page 162.

---

**Web Portal Manager**

To import an existing group from a user registry into the domain and make that group a Tivoli Access Manager group using Web Portal Manager:

1. Log in to the domain.
2. Click **Group + Import Group**.
3. Type a **Group Name** for the group (for example, sales). This field is required.
4. Type a **Registry GID**. This field is required. The registry GID specifies the location in the registry database of the group to be imported. For example: `cn=sales,o=ibm,c=us,dc=mkt`. Lotus Notes users require the full path name for the user being imported. For example: `sales/IBM/US`.
5. Optionally, type the path in the **Object Container field** to the Tivoli Access Manager object space where the group is to be imported.
6. Click **Import**. The new group will be displayed as a link. Select the link and the properties for the new group are displayed.

---

**pdadmin**

To import an existing group from a user registry into the domain and make that group a Tivoli Access Manager group using the `pdadmin` utility, log in to the domain as a domain administrator. Use the `pdadmin group import` command to import an existing group and optionally place this group in a group container object. If the container object does not currently exist, it is automatically created.

For example, to import the existing group named "cn=sales,o=IBM,c=us,dc=mkt" from the user registry, enter:

```
pdadmin sec_master> group import sales "cn=sales,o=IBM,c=us,dc=mkt"
```

**Note:** The group information that is imported to the domain can be imported again to another domain.

For more information, see the IBM Tivoli Access Manager for e-business Command Reference.
Chapter 12. Certificate and password management

This chapter describes how Tivoli Access Manager uses certificates for authentication. Both server-side certificates and client-side certificates are discussed. Additionally, key and stash files are described. Configuration settings for the default lifetime for both the certificates and the key file passwords as well as initial configuration settings are defined.

Note that this chapter describes certificate and password management from an administration C API runtime perspective. However, Tivoli Access Manager also provides a Java runtime for performing the same tasks. See the IBM Tivoli Access Manager for e-business Administration Java Classes Developer Reference and the IBM Tivoli Access Manager for e-business Authorization Java Classes Developer Reference for more information about the administration Java runtime and classes.

This chapter includes:

- “Initial configuration” on page 134
- “Key file and stash file renewal information” on page 135
- “Trust determination” on page 136
- “Server certificate revocation” on page 136
- “Additional key and stash file considerations” on page 137

The Tivoli Access Manager components use Secure Sockets Layer (SSL) for encryption, system authentication, and application-level authentication. SSL uses certificates for operation.

In the secure environment, the policy server acts as the certificate authority (CA) and is responsible for the creation and renewal of certificates. The Access Manager Runtime package (PDRTE) relies on only SSL server-side authentication and does not require a client-side certificate. However, all of the Tivoli Access Manager servers, such as the policy server (pdmgrd), the authorization server (pdacld), the policy proxy server (pdmgrproxyd), and the resource manager servers rely on client-side certificates to operate.

The Tivoli Access Manager servers use certificates to authenticate themselves. For example, when pdacld is communicating with pdmgrd, it presents its client-side certificate. In this example, pdmgrd can be considered the server and pdacld as the client. The pdmgrd server verifies that the certificate is valid and is signed by a trusted signer. In this case, the trusted signer is the pdmgrd itself, using the Tivoli Access Manager certificate authority (PDCA) certificate. The pdacld server does the same for the certificate presented by pdmgrd. As part of the Tivoli Access Manager application-level authentication, after pdmgrd determines that the pdacld certificate is good, it tries to map that certificate to a Tivoli Access Manager user. If the authentication succeeds, then the servers can begin communicating.

The certificates used by Tivoli Access Manager are kept in key files. Key files have a .kdb extension (or .ks extension for Java keystores). These key files should be secured and protected by the strictest operating system controls available because they contain the private keys for the certificates in question. For example, the key file for pdmgrd is ivmgrd.kdb and, by default, it is readable and writeable by only the ivmgr user.
The certificate files in a directory need to be accessible to the policy server user
`ivmgr` (or all users). Make sure that user `ivmgr` (or all users) have permission to
access the `.kdb` file and the directory or folder that contains the `.kdb` file.

Furthermore, to facilitate unattended server operation, there are files that contain
an obfuscated (not encrypted) version of the password to the key files. These
versions are called `stash files`, and the stash files are denoted by a `.sth` file
extension. Note that Java key files generated by Tivoli Access Manager do not have
corresponding stash files. Again, these stash files should be secured using native
standard operating system measures. For `pdmgrd`, the stash file is `ivmgrd.sth` and
its permissions are the same as `ivmgrd.kdb`.

For security reasons, both the certificates and the key file passwords can be set to
expire after a configurable amount of time. The default lifetime for a certificate is
365 days. The default lifetime for a key file password is 183 days. The fixed
lifetime for the PDCA certificate is 20 years. Also by default, the Tivoli Access
Manager components perform self-care by refreshing the certificates and passwords
automatically while they are running. The refresh process reissues a new certificate
with a new lifetime and generates a new password with the configured lifetime.
Note that certificates are not automatically refreshed for Tivoli Access Manager
Java servers.

However, if the servers are not running within a specified window of time, their
certificates or passwords can expire. If this is the case, a manual refresh is
necessary. Furthermore, if a certificate, password or if the entire key file is
corrupted, a manual refresh is warranted to keep the Tivoli Access Manager
domain secure. For information on performing a manual refresh, see “Key file and
stash file renewal information” on page 135.

---

**Initial configuration**

The certificates used by the Tivoli Access Manager components are created as part of
their initial configurations. In a brand-new Tivoli Access Manager installation,
the `pdmgrd` server is the first server configured. As part of its configuration, the
PDCA certificate is created, and a personal certificate that is used by `pdmgrd` is
created and signed by the PDCA certificate. Both of these certificates are located in
the `ivmgrd.kdb` key file. Also, as part of the `pdmgrd` configuration, the Access
Manager Runtime key file `pd.kdb` is created, and the PDCA certificate is inserted
into it as a trusted certificate.

When new systems are added to the Tivoli Access Manager domain, the Access
Manager Runtime package (PDRTE) is configured first. Again, as part of this
configuration, the system `pd.kdb` and `pd.sth` files are created and the PDCA
certificate is included in the key files as a trusted certificate.

When new resource manager servers (such as WebSEAL) are configured, the
`svrssslcfg` tool or equivalent application programming interface (API), is run. This
tool creates a key file (such as `pdac1d.kdb`) and places a personal certificate for the
server in it. The tool also inserts the PDCA certificate as a trusted certificate in the
key file. These two certificates are obtained from `pdmgrd` and are transported to
the client machine over SSL using the Access Manager Runtime key file.

The configuration files and certificate-related stanza entries, such as the configured
key file and the configured stash files, are discussed in “Server
configuration file reference,” on page 231.
Key file and stash file renewal information

The following table lists the components and their associated key files and stash files. It also describes how they are created and refreshed.

Table 4. Component key and stash files

<table>
<thead>
<tr>
<th>Component</th>
<th>Key/Stash File</th>
<th>How it is created</th>
<th>Processes that automatically update the key file, password, or both</th>
<th>Tool for manual update</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDRTE</td>
<td>pd.kdb and pd.sth (does not contain a client-side certificate)</td>
<td>During the PDRTE configuration</td>
<td>Invocations of pdadmin¹</td>
<td>basssslcfg –chgpwd</td>
</tr>
<tr>
<td>pdmgrpxyd</td>
<td>pdmgrproxyd.kdb and pdmgrproxyd.sth</td>
<td>During pdmgrproxyd configuration</td>
<td>Running pdmgr¹</td>
<td>svrsslcfg –chgpwd⁶</td>
</tr>
<tr>
<td>pdacld</td>
<td>ivacld.kdb and ivacld.sth</td>
<td>During pdacld configuration</td>
<td>Running pdacld¹</td>
<td>svrsslcfg –chgpwd⁷</td>
</tr>
<tr>
<td>Resource manager⁶</td>
<td>The key files and stash file names are resource manager-dependent, and the file name is configurable.</td>
<td>Running svrsslcfg –config</td>
<td>Running instance of the resource manager¹</td>
<td>svrsslcfg –chgpwd⁸</td>
</tr>
</tbody>
</table>

Notes:

- ¹ - Automatic certificate and password refresh can be turned off by setting the ssl-auto-refresh stanza entry to no in the [ssl] stanza in the respective configuration file.
- ² - Because pdmgrpxyd also acts as the CA for the secure domain, it must be recycled after a refresh. It continues to operate normally until it is recycled, except it cannot issue or renew certificates for other servers until it is recycled. The policy server log file contains a message stating when the server needs to be restarted.
- ³ - Before running this command, the pdmgrpxyd server must be stopped.
- ⁴ - Before running this command, the pdacld server must be stopped.
- ⁵ - Before running this command, the pdmgrpxyd server must be running, and the pdacld server must be stopped.
- ⁶ - The Java resource managers have an equivalent to key files, where the application personal certificate and the PDCA certificate are stored) but they have no equivalent to stash files. The name of the key files (or Java keystores) are resource-manager dependent and configurable. They key file names are created by running the Java SvrSslICfg class with –action config. For Java keystores, no automatic refresh occurs. To manually update a Java keystore, use the PDAAppSvrConfig.replaceAppSvrCert() method.
- ⁷ - Before running this command, the resource manager must be stopped.
• 8 - Before running this command, the pdmgrd server must be running, and the resource manager must be stopped.

Trust determination

Each of the key files also contains a list of trusted CAs. For Tivoli Access Manager, every key file except for ivmgrd.kdb has the PDCA certificate as a trusted CA. This CA is the certificate that is used to sign all of the other Tivoli Access Manager certificates. This CA is created during pdmgrd configuration and is placed in the ivmgrd.kdb file.

It is extremely important to protect the ivmgrd.kdb file to keep the PDCA certificate’s private key from being compromised. If it is compromised, then it must be regenerated. If this happens, every key file and every certificate in the domain will need to be regenerated as well.

The steps for performing this action are:
1. Regenerate the PDCA certificate (and pdmgrd server certificate) by generating a new ivmgrd.kdb file using mgsslcfg –unconfig and then mgsslcfg –config (pdmgrd must be stopped).
2. Regenerate all Access Manager Runtime (PDRTE) certificates within the domain by first running bassslcfg –unconfig. Next, obtain the CA certificate. You can choose to automatically download the CA certificate, or hand copy a file, depending on the trust you have in your network.
   • If auto-download is on (enabled) and pdmgrd is running, the CA certificate is automatically obtained. By default, auto-download is enabled.
   • If auto-download is off (disabled), the base-64 DER encoded version of the PDCA certificate must be copied to the machine. This file is stored as pdcacert.b64 on the pdmgrd machine.
3. Run bassslcfg –config to complete the Access Manager Runtime (PDRTE) configuration.
4. Regenerate any pdacld key files within the domain by running svrsslcfg –unconfig and svrsslcfg –config (pdmgrd must be running). These commands update both the server certificate for pdacld and its trusted certificate (the new PDCA certificate).
5. Regenerate any other resource manager key file within the domain by running svrsslcfg –unconfig or svrsslcfg –config (pdmgrd must be running). These commands update both the server certificate for pdacld and its trusted certificate (the new PDCA certificate).

From the Java perspective, the Access Manager Runtime also stores the PDCA certificate. If the PDCA certificate is compromised and must be regenerated, all configured Access Manager Runtime (AMJRTEs) must be unconfigured and reconfigured. All resource managers that were previously configured with the SvrSslCfg class must also be unconfigured and then reconfigured.

Server certificate revocation

If a C-based resource manager’s certificate is compromised, you can perform a svrsslcfg –chgcert command to replace the existing server certificate and update the PDCA certificate. For Java-based resource managers, the PDApplSvrConfig.replaceAppSvrCert() method must be used.
Although more difficult, you can also unconfigure and reconfigure the server by running `svrsslcfg --unconfig` and `svrsslcfg --config`. Make sure `pdmgrd` is running. These commands update both the server certificate for `pdacld` and its trusted certificate (the new PDCA certificate). Similarly, a Java-based resource manager can be unconfigured and reconfigured using the Java `SvrSslCfg` class.

### Additional key and stash file considerations

Additional considerations for key file and stash file renewal include:

- When a certificate and the password to the key file containing that certificate are both expired, the password must be refreshed first. For example, for `pdacld`, run `svrsslcfg --chgpwd` and then `svrsslcfg --chgcert`. This is necessary because a valid password is needed to open the key file to get to the certificate.

- The value for the lifetime of a certificate is controlled by the value of the `ivmgrd.conf`, `[ssl]` stanza, and `ssl-cert-life` stanza entry attribute when `pdmgrd` is started. Any certificates issued or renewed use this value. To increase or decrease this value, change the value and restart `pdmgrd`. The new value is in effect only for certificates that are issued or renewed from that point onward. The actual time used will be the lesser of the value specified in the `ivmgrd.conf` configuration file and the number of days before the policy server CA certificate expires.

- For automatic password renewal, the value for the lifetime of a password is controlled by the value of the `[ssl]`, `ssl-pwd-life` attribute in effect when the server is started. For manual password renewal, the value is dictated by the value supplied to the `--chgpwd` command. This value is also written into the appropriate configuration file.

- Tivoli Access Manager servers can also communicate with Lightweight Directory Access Protocol (LDAP) using SSL. In the standard configuration, this communication uses server-side authentication only. Therefore, the Tivoli Access Manager server needs only the CA certificate that signed the LDAP server certificate or the LDAP server certificate itself. The expiration and management of these certificates are not handled by Tivoli Access Manager. However, it is possible to include the LDAP certificate in the key file for a resource manager by running `svrsslcfg --config` and using the `--C` option.

For certificates that are not managed by Tivoli Access Manager, these certificates must be refreshed by using the same mechanism used to create the initial certificate. The new certificate can be replaced in the key file by running the `svrsslcfg --modify --C new_cert_filename` command.

- After running `basslcfg --config`, it might be necessary to change the permissions of `pd.kdb` and `pd.sth`.

- The configuration files mentioned are generally found in the `install_dir/etc` directory. For example, on AIX the `pdmgrd`, `pdacld`, and `runtime` configuration files are found in `/opt/PolicyDirector/etc/ivmgrd.conf`, `/opt/PolicyDirector/etc/pdacld.conf`, and `/opt/PolicyDirector/etc/pd.conf` respectively. Similarly, the key files and stash files can be found in the `install_dir/keytabs` directory.

- Tivoli Access Manager does not distinguish between export and domestic encryption. For Java-based encryption, the strength is regulated by the jurisdiction files that are present in the Java runtime environment. There is no set length for keys generated by the Access Manager Runtime.

- Both the public keys that are included in certificates and the private keys that might be stored in key files have key lengths. The maximum key length is 2048 bits. Public keys having 2048 bits key lengths can be generated by using the configuration utilities (`basslcfg`, `mgrsslcfg`, or `svrsslcfg`).

Chapter 12. Certificate and password management 137
Chapter 13. Server management

This chapter provides detailed information for performing general administration and configuration tasks on the Tivoli Access Manager servers.

This chapter contains the following sections:
- “Tivoli Access Manager servers”
- “Tivoli Access Manager utilities” on page 142
- “Tivoli Access Manager servers tasks” on page 142
- “Server configuration file tasks” on page 144
- “Policy server administration tasks” on page 145

Tivoli Access Manager servers

Tivoli Access Manager consists of the following server processes, or daemons:
- Policy server (pdmgrd)
- Authorization server (pdacld)
- Proxy server (pdmgrproxyd)

The policy server (pdmgrd) manages the policy database, referred to as its original name of master authorization database, and maintains location information about other Tivoli Access Manager servers in the domain. There must be at least one policy server defined for each domain.

The authorization server (pdacld) allows other applications to make authorization calls, using the authorization application programming interface (authorization API), to Tivoli Access Manager. The authorization server also acts as a logging and auditing collection server to store records of server activity.

The proxy server (pdmgrproxyd) helps support several network deployment strategies for the policy server and the resource managers. A resource manager can...
be any server or application that uses the Authorization API to process client requests for access to resources, such as WebSEAL servers or Authorization API applications.

Proxy server

A proxy server is a server that acts as an intermediary between a less trusted network and a more trusted network so that the enterprise can ensure security, administrative control, and caching service. A proxy server is associated with or part of a gateway server that separates the enterprise network from the outside network and a firewall server that protects the enterprise network from outside intrusion. In a Tivoli Access Manager environment, the proxy server runs on behalf of the policy server for a given number of resource manager and administrative functions, such as the pdadmin commands.

The proxy server serves many important functions in a Tivoli Access Manager environment. The proxy can be used to terminate any connections from a less trusted network and to pass those requests to a policy server in a more trusted network using a different connection. This protects the policy server in the more trusted network from denial-of-service attacks and other similar attacks. In this deployment scenario the proxy is deployed in what is commonly called the demilitarized zone (DMZ).

Also, the proxy is useful in a wide-area network deployment where the policy server and several applications are deployed at separate locations across a slow connection. Typically this happens when the policy server and the applications are deployed in different geographical locations. If a proxy is deployed on the same network as the applications, and the applications are configured to go through the proxy, then only the proxy will be contacting the policy server instead of each application. This is important in several respects.

- The proxy server can be configured to cache security policy such that when a policy update occurs at the policy server, only one copy of the policy is transmitted from the policy server to the proxy. The proxy then provides the policy to all of the applications. If the proxy was not there, each individual application would request and receive the policy from the policy server, significantly increasing the network traffic.
- This configuration can also improve security because firewalls between the locations can be configured to only allow the proxy to contact the policy server and not the applications.

Figure 23 on page 141 shows the interaction between applications, the proxy server, and the policy server.
Server dependencies

Make sure you take the following dependencies into account for your server configuration:

- There must be at least one instance of the policy server.
- There must be at least one policy server defined. You can have a single policy server and create as many domains as you want. When a domain is created, a separate policy database is also created for each domain. The single policy server can access any of the distinct policy databases.
- The policy server (pdmgrd) manages the policy database.
- There must be only one policy database (master authorization database) in a domain.
- The policy database should reside on a highly available pdmgrd server with a robust file system.
- It is strongly recommended that each policy database be subject to a regular backup procedure. The administrator can specify the location for the backup files.
- The policy servers provide authorization database replication services to all other Tivoli Access Manager servers in the domain running in local cache mode.
- Each resource manager, such as Tivoli Access Manager WebSEAL, Tivoli Access Manager for Business Integration, or Tivoli Access Manager for Operating
Systems, applies security policy based on information from either the policy database or from a replicated authorization database.

**Tivoli Access Manager utilities**

The Tivoli Access Manager utilities are discussed in detail in the *IBM Tivoli Access Manager for e-business Command Reference*. The table at the beginning of the utilities section lists available utilities and their purposes.

Note that the `pdadmin` command line interface, which is also discussed in *IBM Tivoli Access Manager for e-business Command Reference*, provides commands that assist in troubleshooting problems. For example, the `pdadmin` command, includes `server task stats` and `server task trace` options that let you turn on statistics gathering and capture information about error conditions. In addition, the *IBM Tivoli Access Manager for e-business Problem Determination Guide* provides further diagnostic information for using the Tivoli Access Manager `pdadmin` command and other utilities.

**Tivoli Access Manager servers tasks**

This section describes how to start and stop server processes:

- "Start and stop servers on UNIX systems"
- "Start and stop servers on Windows systems” on page 143

### Start and stop servers on UNIX systems

Server processes are normally enabled and disabled through automated scripts that run at system startup and shutdown.

In a UNIX environment, you can also use the `pd_start` script to manually start and stop the server processes. This technique is useful when you need to customize an installation or when you need to perform troubleshooting tasks. You can run scripts only on the local machine.

The general syntax for `pd_start` is as follows:

```
# pd_start {start|restart|stop|status}
```

You can run the `pd_start` utility from any directory. The script is located in the following directory:

```
/opt/PolicyDirector/bin/
```

### Start the Tivoli Access Manager servers using the `pd_start` utility

Use the `pd_start` utility to start all Tivoli Access Manager servers not currently running on a particular machine:

```
# pd_start start
```

This script waits until all servers have started before returning the prompt.

### Start individual servers manually

You can manually start the servers individually by executing the server directly.

You must perform the startup commands as an administration user, such as `root`.

Start the Tivoli Access Manager servers in the following order:

1. For the policy server (`pdmgrd`), enter the following:
install_path/bin/pdmgrd

2. For the policy proxy server (pdmgrproxyd), enter the following:
   install_path/bin/pdmgrproxyd

3. For the authorization server (pdacld), enter the following:
   install_path/bin/pdacld

Restart the Tivoli Access Manager servers using the pd_start utility
Use the pd_start utility to stop all Tivoli Access Manager servers on a particular machine and then restart the servers:
   pd_start restart

This script waits until all servers have started before returning the prompt.

Stop the Tivoli Access Manager servers using the pd_start utility
Use the pd_start utility to stop all Tivoli Access Manager servers on a particular machine in the correct order:
   pd_start stop

This script waits until all servers have stopped before returning the prompt.

Display server status using the pd_start utility
Use the pd_start command to display server status:
   pd_start status

<table>
<thead>
<tr>
<th>Tivoli Access Manager Servers:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>Enabled</td>
<td>Running</td>
</tr>
<tr>
<td>pdmgrd</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>webseald</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>pdacld</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>pdmgrproxyd</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

Start and stop servers on Windows systems
On Microsoft Windows systems, use the Services window from the Control Panel to start and stop the server processes manually. This can be useful when customizing an installation or when troubleshooting. Administrative privileges are required to use this utility.

You can start and stop the Tivoli Access Manager servers all at once or individually. The servers generally must be stopped and started in the correct order.

Use the Services Control Panel to stop and start servers
The AutoStart Service automatically starts each of the Tivoli Access Manager servers whenever the Startup configuration is set to Automatic. After the servers start, the AutoStart Service exits.

You can also use the Services Control Panel to manually start and stop the individual servers:
1. Open the Windows Control Panel.
2. Double-click Administrative tools, and the administrative tools folder opens.
3. Double-click the Services icon.
   The Services dialog appears.
4. From the list box, select the Tivoli Access Manager servers according to the sequence indicated in Steps 5 and 6.

5. **Stop** the Tivoli Access Manager servers in the following order:
   - authorization server
   - proxy server
   - policy server

6. **Start** the Tivoli Access Manager servers in the following order:
   - policy server
   - proxy server
   - authorization server

7. Click the appropriate **Startup Type** control option button (**Start, Stop, Restart**) from the right-hand side of the box.

8. To prevent automatic starting of a Tivoli Access Manager server by the AutoStart Service, use the startup properties to set that server **Startup Type** to Disabled.

---

**Server configuration file tasks**

You can use the server configuration files to customize the operation of Tivoli Access Manager and its servers. Various server configurations are discussed in Appendix A, “Server configuration file reference,” on page 231.

**Change configuration settings**

The configuration files, stanzas, and stanza entries are described in Appendix A, “Server configuration file reference,” on page 231.

To change a server configuration setting, do the following:

1. Make a backup copy of the configuration file you plan to modify.
   This allows you to return the configuration file to a known working state, should you encounter an error later.

2. Stop the Tivoli Access Manager servers affected.

3. Make the changes by doing one of the following:
   - Use an ASCII text editor to edit the configuration file and make any necessary changes. Save your changes.
   - Use the **pdadmin config** commands to modify the configuration file.
   - Use the appropriate configuration tool for your server to change the configuration settings:
     - For ivmgrd.conf, use the **mgrsslcfg** utility.
     - For pd.conf, use the **basslcfg** utility.
     - For all other configuration files, use the **svrsslcfg** utility.
   - Many stanzas or values are created or modified only by using Tivoli Access Manager configuration utilities. Some values are filled in automatically after the configuration is completed.

   **Note:** Do not manually edit these values.

4. Start the Tivoli Access Manager servers affected.

   For example, if you want to change the ivmgrd.conf file, you must stop the policy servers, make the change, and then restart all of the policy servers for the change to become effective.

---

IBM Tivoli Access Manager: Base Administration Guide
Automate server startup at boot time

Stanza entries for automating server startup are located in the [pdrte] stanza of the pd.conf configuration file.

By default, the pd.conf file is installed at the following location for UNIX:
/opt/PolicyDirector/etc/pd.conf

By default, the pd.conf file is installed at the following location for Windows:
c:\Program files\tivoli\Policy Director\etc\pd.conf

Policy server
When the PDMgr package is installed, the policy server automatically starts after each system reboots:

[pdrte]
boot-start-ivmgrd = yes

To prevent automatic pdmgrd startup, set:
boot-start-ivmgrd = no

Authorization server
When the PDACld package is installed, the authorization server daemon automatically starts after each system reboots:

[pdrte]
boot-start-ivacld = yes

To prevent automatic pdacld startup, set:
boot-start-ivacld = no

Proxy server
When the PDMgrProxyd package is installed, the proxy server daemon automatically starts after each system reboots:

[pdrte]
boot-start-pdmgrproxyd = yes

To prevent automatic pdmgrproxyd startup, set:
boot-start-pdmgrproxyd = no

Policy server administration tasks

The policy server manages the policy database (or databases), and maintains location information about other Tivoli Access Manager servers in each domain. The policy server typically requires very little administration or configuration. This section describes configuration tasks available to the administrator.

- “Replicate the authorization database” on page 145
- “Set the number of update-notifier threads” on page 147
- “Set the notification delay time” on page 147

Replicate the authorization database

A Tivoli Access Manager domain administrator can make security policy changes to a domain at any time. A primary responsibility of the policy server is to make the necessary adjustments to the domain master authorization database to reflect these changes.
When the policy server makes a change to the master authorization database, it can send out notification of this change to all resource manager servers (with replica databases). The authorization servers must then request a database update from the policy server.

**Note:** Additionally, resource manager servers can check for database updates by polling the policy server at regular intervals. Polling configuration for a WebSEAL client, for example, is explained in the *IBM Tivoli Access Manager for e-business WebSEAL Administration Guide.*

Tivoli Access Manager allows you to configure update notifications from the policy server to be an automatic process or a manually controlled task. The `auto-database-update-notify` stanza entry is located in the `[ivmgrd]` stanza of the `ivmgrd.conf` configuration file. By default, the stanza entry value is set to `yes` (update notification is automatically performed by the policy server):

```
[ivmgrd]
auto-database-update-notify = yes
```

This automatic setting is appropriate for environments where database changes are few and infrequent. When you configure update notification to be automatic, you must also correctly configure the `max-notifier-threads` and `notifier-wait-time` stanza entries. For more information on `max-notifier-threads` and `notifier-wait-time` stanza entries, see “Set the number of update-notifier threads” on page 147 and “Set the notification delay time” on page 147.

When you configure update notification to be manual, manual application of the `server replicate` command controls this event.

```
[ivmgrd]
auto-database-update-notify = no
```

This manual setting is appropriate for environments where database modifications occur frequently and involve substantial changes. In some cases several database modifications can generate many update notifications that soon become obsolete because of the continuing changes to the master database. These obsolete notifications cause unnecessary network traffic and impair the performance of resource managers because of continued requesting and processing of policy updates.

The manual control of update notification allows you to complete the process of modifying the master authorization database before update notifications are sent out to authorization servers with database replicas.

In manual mode, update notification uses the notifier thread pool (as it does in automatic mode). Therefore, the manual mode setting is affected by the `max-notifier-threads` stanza entry value. For more information on the `max-notifier-threads` stanza entry, see “Set the number of update-notifier threads” on page 147.

**Using the pdadmin server replicate command**

When you configure update notification to be manual, manual application of the `server replicate` command controls this event.

```
padmin_secmaster> server replicate -server test_server
```

If the optional `server-name` argument (test_server) is specified, only that server is notified of changes to the master authorization database. A response is returned indicating the success or failure of the notification and the replication.
If the **server-name** argument is not specified, all configured resource manager servers receive update notifications. A successful response indicates only that the policy server has begun sending out update notifications. The response does not indicate success or failure of the actual notification and replication processes.

The authorization required to execute this command is the **s** action bit on the /Management/Server object.

For more information about the **server replicate** command, see the *IBM Tivoli Access Manager for e-business Command Reference*.

**Set the number of update-notifier threads**

The policy server is responsible for synchronizing all database replicas in the domain. When a change is made to the master database, notification threads do the work of announcing this change to all replicas. Each replica then has the responsibility to download the new information from the master.

The policy server configuration file, ivmgrd.conf, contains a stanza entry for setting the maximum number of update-notifier threads. This pool of threads allows simultaneous (parallel) notification.

For example, to concurrently notify 30 replicas of a database change, the thread pool should be set to at least 30. If there are more than 30 replicas, another round of notifications occurs (in this example, 30 at a time). All replicas are guaranteed to be notified, regardless of the value of this stanza entry.

The performance goal of the update-notifier threads value is to announce a database change as quickly as possible. Generally the value should be set to equal the number of existing replicas. This results in the performance advantage of a single pool of threads quickly accomplishing the notification task to all replicas at once.

The default event notifier thread pool is set as the following:

```
[ivmgrd]
max-notifier-threads = 10
```

When the auto-database-update-notify stanza entry is set to yes, you must correctly configure this stanza entry and also the notifier-wait-time stanza entry. See also "Set the notification delay time."

**Set the notification delay time**

When the policy server is instructed to make a change to the master authorization database, it waits for a default period of time before sending out notifications to database replicas. The default time delay is set at 15 seconds. This time delay is reset with each subsequent change to the database.

The purpose of the time delay is to prevent the policy server from sending individual replica notifications for each in a series of database changes. The time delay helps to ensure optimal performance of the Tivoli Access Manager system.

This performance feature is particularly important for environments where batch changes are made to the authorization database. It is not efficient for policy changes to be sent to database replicas until all changes have been made.
You can override this default notification time delay by changing the
notifier-wait-time stanza entry value (in seconds), located in the [ivmgrd] stanza
of the ivmgrd.conf configuration file. For example:

```
[ivmgrd]
notifier-wait-time = 20
```

By default, the value is set to 15 seconds.

When the auto-database-update-notify stanza entry is set to yes, you must
correctly configure this stanza entry and also the max-notifier-threads stanza
entry. See also “Set the number of update-notifier threads” on page 147.
Chapter 14. High availability of policy server

This chapter provides information on ensuring that Tivoli Access Manager provides high availability for the policy server should a server failure occur. This chapter also describes how Tivoli Access Manager supports the replication capability of the LDAP directory to ensure that the directory data is always available.

This chapter includes information about:

- “Data integrity” on page 150
- “Primary and replica LDAP servers”
- “Active and passive policy servers”
- “High availability management” on page 150

Data integrity

You should ensure that the data that is needed by Tivoli Access Manager is always available. To ensure data redundancy, all your data should be stored on data devices that are Redundant Array of Independent Disks (RAID) secured.

Authorization information and decision making can be off-loaded to authorization servers. All data also should be subject to a robust backup process to ensure that you can recover the data in the event of a hardware or software error. The pdbbackup utility provides backup, restore, and extract capabilities for Tivoli Access Manager data. See IBM Tivoli Access Manager for e-business Command Reference for more information on this utility.

Primary and replica LDAP servers

Tivoli Access Manager allows primary and replica LDAP servers. The replica LDAP server, on a different node, can assume LDAP server operations if the primary LDAP server fails.

It is important to note that, during failover, no write operations can occur. Only read-only LDAP server operations are permitted during failover.

Refer to the LDAP server documentation for complete information about high availability of LDAP servers.

Active and passive policy servers

The policy server manages the master policy database as well as the policy databases that are associated with other secure domains that you might create. The policy server also maintains location information about other servers in the domain. When the policy server (pdmgrd) fails or when the system on which the policy server is located become unavailable, an outage might result because of no data redundancy.

To provide the redundancy for the shared data and for the functions that are provided by the Tivoli Access Manager policy server, you can install and configure a primary policy server and a standby policy server. The standby server takes over policy server functions in the event of a system or primary policy server failure.
The standby policy server acts as the primary policy server until the original primary policy server is up and running again with the standby server back to serving as the failover server.

If you plan to set up a primary and standby policy server, these rules apply:

- A 2-node IBM AIX High-Availability Cluster Multiprocessing (HACMP) environment consisting of one active server and one standby server has been tested. Therefore, IBM supports only a IBM AIX HACMP version 4.4.1 or later 2-node environment for this release.
- A primary and a standby policy servers must be installed and configured on separate machines, and both policy servers must be within the AIX HACMP cluster environment.
- The user registry servers (such as the LDAP server, Microsoft Active Directory, or IBM Lotus Domino) must be on a machine other than the machines on which the primary and standby policy servers are installed.
- It is important to back up any shared data or any shared policy databases before you begin to configure the primary and standby servers to the shared file system.
- Each AIX system must have access to a shared disk array that is configured for data redundancy.
- The primary and a standby policy server must be configured to the shared file system, and the shared file system must be mountable by each server.
- Both the policy database and the configuration files, which are used by the policy server, must be located on a shared disk array.

Follow the procedure in the *IBM Tivoli Access Manager Base Installation Guide* for setting up a policy standby server.

---

### High availability management

The procedure for setting up a policy standby sever is discussed in the *IBM Tivoli Access Manager Base Installation Guide*. The following tasks are procedures to ensure that you correctly followed the initial Tivoli Access Manager configuration procedures for setting up HACMP Tivoli Access Manager primary and standby servers.

### Verify the high-availability policy servers setup

To verify that the installation and configuration procedures were correctly followed, ensure that these three main tasks have been completed:

1. Make sure that you have set up the required soft links from the active primary server to the standby server.
2. Make sure that you have modified the appropriate configuration options in the ivmgrd.conf and pd.conf configuration files on both the primary and standby policy servers and that the configuration files have the same default settings for the required user and group IDs:
   - ivmgr user ID
   - ivmgr group ID
   - tivoli user ID
   - tivoli group ID
3. Make sure you copy files from the local AIX file system for the primary server, the standby server, and the policy server to the shared file system, that the shared file system is located on a common directory, and that the each user and group has the necessary access permissions.
If any of these items are incorrectly set, return to the procedure for setting up a standby policy server in the *IBM Tivoli Access Manager Base Installation Guide* to ensure the procedure was correctly followed.

**Review log files**

You can monitor the transition process of primary policy to the standby server by examining the hacmp.log file to verify that all HACMP failover operations occurred. The procedure for reviewing HACMP logs can be found in the in the HACMP documentation. This log is usually found in the /tmp directory.

If any Read or Write operation errors occurred during policy server failover, you can review the primary policy server log files. The location of the Tivoli Access Manager log files is dependent upon whether or not Tivoli Common Directory is used. Refer to Chapter 17, “Log and routing files,” on page 169 for more information about Tivoli Access Manager log files. The *IBM Tivoli Access Manager for e-business Problem Determination Guide* also describes log files and the XML log file viewer.
Chapter 15. Multiple-tenancy policy server

A multiple-tenancy server refers to a server that permits the hosting of multiple customers on a single server instead of on multiple client machines. For example, your company might be sharing applications or data on your company’s server with your customer (for example, Smith-Davis Enterprises). Before you add data and information that belongs to another customer (for example, Systems, Inc.), you must somehow ensure that these two customers cannot get access to the other company’s data or applications.

Using a multiple-tenancy (multi-domain) server, you can run each company’s applications or data in an isolated server environment. Running in an isolated or partitioned server environment replaces the need to use multiple physical servers for each customer and their applications. Depending on the demands of your customers and their applications, you can host multiple clients on a single server. Replacing multiple servers with one server reduces the costs to your company for the services you provide to your customers. For example, fewer servers reduce your hardware costs, reduce your IT personnel burden, and a single server is easier to manage than are multiple servers.

A multiple tenancy server does not have to be less secure than the traditional one-server, one-client approach. Using technologies such as SSL and restricted access, you can protect two customers (users) on the same server from one another. Extra layers of security for multiple-user applications have been designed into Tivoli Access Manager. Tivoli Access Manager compartmentalizes each domain to seal users off from one another rather than using the multiple-user security provisions of UNIX or of Windows NT.

The Tivoli Access Manager runtime clients must be configured into a specific domain at installation time. The domain membership information accompanies each subsequent request from the client to the policy server. The [domains] stanza in the ivmgrd.conf configuration file for the multiple-tenancy policy server (pdmgrd) contains a list of valid existing domains. See “[domains] and [domain=domain_name] stanzas” on page 268 for an explanation of each stanza entry.

Each domain must have its own [domain=domain_name] stanza. For example, to set up separate domains for Smith-Davis Enterprises and Systems, Inc., you might create two domains uniquely named smithdavis and systemsinc, respectively:

[domains]

    domain = smithdavis
    domain = systemsinc

[domain=smithdavis]

[domain=systemsinc]

The multi-tenancy domains implemented by Tivoli Access Manager results in separate databases for each protected object space. All the databases can use the same underlying user registry (one LDAP registry with distinct and separate distinguished names). For example, if you wanted to specify the sde0001.db file, you would specify the file name and directory in this stanza entry:
[domain=smithdavis]
database-path = D:\smithdavis\sde0001.db

The distinguished name (DN) restricts into which registry the users can be created in or imported into. The distinguished name substrings must appear in the user's distinguished name, for example:
cn=sdeuser1,ou=sde,dc=mkt,o=US

Representing the following:
  cn = user name (sdeuser1) for K.L. Logan
  ou = company/organization (sde) for Smith-Davis Enterprises
  dc = division (mkt) for Marketing Group
  o = country (US) for United States

If you wanted to restrict user accounts to be created in the dc=mkt,o=US directory container for the smithdavis domain, you would specify to allow this registry substring in this stanza entry:
[domain=smithdavis]
allowed-registry-substrings = "dc=mkt,o=US"

Or, if you wanted to restrict user accounts to be created in the dc=mkt directory container for the smithdavis domain, regardless of where that container exists within the registry, you would specify:
[domain=smithdavis]
allowed-registry-substrings = "dc=mkt"

A completed [domains] stanza in the ivmgrd.conf configuration file for the policy server (pdmgrd) might look like this:
[domains]
domain = smithdavis
domain = systemsinc

[domain=smithdavis]
database-path = D:\smithdavis\sde0001.db
allowed-registry-substrings = "dc=mkt,o=US"

[domain = systemsinc]
database-path = D:\systemsinc\sysinc0001.db
allowed-registry-substrings = "dc=sales,o=US"
Chapter 16. Delegated administration

Tivoli Access Manager allows high-level administrators to delegate responsibilities for managing the domain to lower-level administrators. This capability is vital to successfully managing very large domains composed of numerous departments.

Tivoli Access Manager supports delegated administration in the following areas:

- Delegated management of resources in subregions of the object space
  Administration capabilities are restricted to a portion of the object space.
- Delegated management of groups and users
  Administration capabilities are restricted to a portion of the user population.

This chapter contains the following sections:

- “Overview of delegate administration”
- “Delegated role administration” on page 157
- “Delegated object space management” on page 158
- “Delegated user and group management” on page 160
- “Delegated administration security policy” on page 166

Overview of delegate administration

Delegate administration provides a Tivoli Access Manager administrator the capability to create delegate user domains, create new users, add existing users to additional domains, and assign various types of administrators to the domains. These delegate administrators can then perform a subset of administration functions, depending on their type, on the users in their assigned domain. This concept of delegate user administration can be applied to all Tivoli Access Manager users so that a hierarchy of user domains is formed. In this hierarchical arrangement, each Tivoli Access Manager user can be managed only by the administrators for the domain of which the user is member or by the administrators for the super domains (explained later in this chapter). The actual functions that administrators can perform depend on their assigned administrator type.

A Tivoli Access Manager administrator, such as sec_master, can create a number of enterprise domains and assign one or multiple types of administrators to each enterprise domain. The administrator for an enterprise domain can create new users in the domain and add existing Tivoli Access Manager users to the domain.

In addition to this user-related function, Tivoli Access Manager administrators can create new domains below the enterprise domain level (subdomains) and assign users to be the administrators for these new domains (domain administrators). Administrators of the new domains can then create new users in their own domain.

The Tivoli Access Manager administrator for the enterprise domain (the domain’s superdomain) also has authority to administer the domain. Tivoli Access Manager administrators can create and manage as many domains under their authority as necessary to fulfill their unique business needs.
Note: An enterprise domain is basically the top-level domain, and any domain created below an enterprise domain level is just called a domain.

As an example of this type of multiple domain administration in Figure 24, a Tivoli Access Manager administrator can create enterprise domains A and B and assign an administrator for each domain. The domain administrator for enterprise domain B can create new users P, Q. A Tivoli Access Manager administrator can create domains C and D below the enterprise domains A and B, and assign domain administrators to C and D. The Tivoli Access Manager administrator can then create domain E below domain D, and assign a domain administrator to E. The domain administrator for domain E can then create new users X, Y, and Z within domain E. Because a domain administrator for a domain can also administer that domain’s subdomains, both the domain administrators for domain D and the domain administrator for enterprise domain B can create users (or perform other administrative functions) for domain E.

![Diagram of domain hierarchy]

Figure 24. Delegate administrators

For each delegate user domain (including the enterprise domain), predefined administrator types can be assigned in that domain. The following are the various administrator types and the set of administrative functions that can be performed by administrators assigned to each of these types:

- **Tivoli Access Manager Administrator.** The Tivoli Access Manager administrator is a member of the iv-admin group. The Tivoli Access Manager administrator can perform all delegate administration functions.

- **Domain Administrator.** The domain administrator can perform administrative functions for the users in their domain. Domain administrators can create new users and administrators in their own domain, and assign an existing domain user to be an administrator (of any type except domain administrator) for the domain.

- **Senior Administrator.** A senior administrator has the same authority as a domain administrator, except that a senior administrator cannot assign additional administrators.
• **Administrator.** An administrator has the same authority as a senior administrator, except that an administrator cannot create new domain users. An administrator can modify an existing user’s properties.

• **Support Administrator.** A support administrator serves the user in a help desk role and is able to view users’ properties, change users’ passwords, and modify the Is Password Valid? flags for users.

The delegate user administration tool enforces the administrative functions that can be performed with each administrator type. When an administrator logs in, administrative functions become available in accordance with that user’s administrator type.

## Delegated role administration

Another part of the Tivoli Access Manager delegate administration system is role administration. To successfully deploy Tivoli Access Manager, a security policy must be defined that regulates access to objects, and the actions that can be performed on those objects. Execution of this policy is usually difficult because the security policy is often defined by high-level members of an organization with an emphasis on global security issues. The policy then must be put into action by local members of the organization, who see the lower-level details and implementation concerns. Often these two groups have similar goals for overall organizational security, but interconnecting these two disparate points of view is challenging. Role-based administration provides an enhanced ability for organizational security to meet the requirements of today’s complex security requirements for scalability, simplicity, and flexibility.

To understand role administration, the first concept that must be defined is a role. A role consists of a number of tasks, responsibilities, or skills required to fulfill a specific job requirement. When this definition is contrasted against the access control list (ACL) model of Tivoli Access Manager, a role becomes a list of one or more pairs of objects and one or more access permissions that are applied to the object. For example:

- object 1: permission 1
- object 2: permission 2, 3, and 4
- object 3: permission 5

In order for a role to be used it must be activated. A role is activated when an Tivoli Access Manager administrator enables its definition in the Tivoli Access Manager namespace. After a role is activated and a user is assigned to the role, the user has permission 1 for object 1, permission 2, 3, and 4 for object 2, and permission 5 for object 3. The access permissions for these objects allow the user to access the objects, and therefore perform the job responsibility defined by the role. For example, an accountant role can be defined to consist of the following two pairs of objects and permissions:

- Payroll check object: create/modify/delete
- Reimbursement request object: approve

When this role is activated and an employee in the accounting department is assigned to this role, that employee is able to create, modify, or delete a payroll check and approve a reimbursement request; thus, performing the job that an accountant is expected to perform.
To successfully administer roles, an administrator must be able to perform three types of tasks:

- Role creation
- Role assignment
- Role activation

*Role creation* involves defining a role so that it has a list of one or more pairs of Tivoli Access Manager objects and permissions that can be applied to the objects. When a role is created, a Tivoli Access Manager group is created to represent the role. A corresponding group object in the management object space is also created. The object/permissions pair information for the role is stored in the extended attributes associated with the group object. Only a Tivoli Access Manager administrator is able to create a role.

*Role assignment* consists of assigning a user to a role that has already been created. The purpose behind assigning users to roles is to let those users have the access permissions on the objects defined in the role. This function reduces the workload involved in maintaining user-permission-object relationships, because role assignment is separated from object/access permission management. When a user is assigned to a role in Web Portal Manager, the user is added as a member of the group that represents the role. Domain administrators, senior administrators, and administrators of a domain can assign users in their domains to a role.

*Role activation* enables a newly created role to function. After a role is created and a user is assigned to that role, the user does not have access permissions for the objects defined in the role until the role is activated. When a role is activated in Web Portal Manager, an ACL entry that contains the group that represents the role and the access permissions defined in the role are added to the ACL for each object defined in the role. Because a user has been added to the group when the user is assigned to the role, that user has permissions to access the objects only after a role is activated. Only a Tivoli Access Manager administrator is able to activate a role.

A role is an entity that can be delegated and administered. When a role is created, it can be assigned to an enterprise domain. Domain administrators can in turn assign any of the roles within that domain to any subdomain. When a role is assigned to a subdomain, an administrator for that subdomain can assign any subdomain users to that role. This process of assigning roles to subdomains can be repeated as needed so that roles can be made available to the appropriate users. Role assignment to an enterprise domain can be performed only by the Tivoli Access Manager administrator. Domain administrators can assign a role to their subdomains.

### Delegated object space management

The distribution of administration responsibilities within a domain is called *management delegation*. The need for management delegation generally arises from the growing demands of a large site containing many distinct departmental or resource divisions.

Typically, a large object space can be organized into regions representing these departments or divisions. Each distinct region of the domain is usually better organized and maintained by a manager who is more familiar with the issues and needs of that branch.
Structuring the object space for management delegation

Structure your object space to contain distinct regions, or branches, where submanagement responsibilities specific to that branch can be carried out.

In Figure 25 both the Engineering and Publications regions of the object space require separate management control. Control of these regions begins with the root of each region and extends to all objects below the root.

![Diagram of object space](image)

Figure 25. Structuring the object space for management delegation

Default administration users and groups

Tivoli Access Manager provides several important administration groups during installation. For information on these user and groups, see “Default administration users and groups” on page 43.

Example: Management delegation

A large object space might require many administration users to manage a variety of subbranches. In this scenario, the access control lists (ACLs) for the directories on the path to each of these branches must contain entries for each account, with traverse permission. For a site with many administration users, these ACLs could contain a long list of entries representing all these administration accounts.

The following technique resolves the problem of numerous ACL entries for administrators:

1. Create an administration group account.
2. Add all new administration users to this group.
3. Add this group as an ACL entry (with traverse) to the directories leading to each subbranch that requires management delegation.
4. At each branch root ACL, create an administration group for each subbranch and add the appropriate user to the appropriate subbranch administration group (with b, c, T, plus other appropriate permissions).
5. The administrator can now remove the administration group ACL entry (and any other entry) from the root.

Now, only that user has control over the root and all objects below the root.

In Figure 26 on page 160 the group iv-admin contains all administration users. User pub-manager is a member of this group and therefore, has the necessary traverse permission required to navigate to the Publications directory.
The Publications directory includes the user pub-manager entry in its ACL. Because pub-manager is the delegated administrator of this branch (with the appropriate permissions), pub-manager can remove the iv-admin group account (and any other ACL entries) from the Publications ACL to gain total control over that branch of the Web space.

Delegated user and group management

In order to manage a large or complex set of users, you can delegate the management of specific groups of users to lower-level administrators. When an administrator is given policy management control of a group, that administrator has policy management control over the user members of that group.

Delegated group management defines:
- Who has administration responsibility for a specific group (and the user members of that group).
- What level of group and user control has been given to this administrator.

In this discussion, the term, administrator, refers to the responsibilities and controls granted to an otherwise typical user. An administrator of delegated duties is a normal user with additional powers to perform certain management tasks.

Setting up delegated group management requires the following steps:
1. Determine a logical and practical hierarchy of the users and user types who are members of the domain.
2. Create group container objects that reflect this hierarchy.
3. Create appropriate administration groups within these container objects.
4. Add the appropriate user to the appropriate administration group with the specific permissions needed to perform the required tasks.
Create group container objects

By default, the /Management region of the Tivoli Access Manager object space has a Groups container object that you can use to organize the hierarchy of groups in your domain.

Container objects are structural designations that allow you to organize the object space into distinct and hierarchical functional regions. Group container objects allow you to define distinct categories of group types.

To create actual groups within each specific group container object using the Web Portal Manager or the pdadmin command line utility, log in to the desired domain as a domain administrator.

Web Portal Manager

To create a new group container object using Web Portal Manager:

1. Log in to the domain.
2. Click Object Space → Create Object.
3. Type an Object Name. This field is required. Type the full path for the object name. For example: /Management/Groups/Travel
4. Type a Description for the object space. For example: Travel Container Object
5. Click Create. To see the new object in the hierarchical structure, browse the object space. See "Browse the object space" on page 67.

![Group container object diagram](image)

Figure 27. Group container object

pdadmin

To create a new group container object using the pdadmin utility, log in to the domain and use the pdadmin object create command.

For example, to create a new group container object:

```
pdadmin> object create obj-name description type
   ispolicyattachable {yes|no}
```

Table 5. Parameters for creating an object

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj-name</td>
<td>Full path and name of the new group container object. Path must begin with /Management/Groups.</td>
</tr>
<tr>
<td>description</td>
<td>Any text string describing the object. This information appears in the object show command.</td>
</tr>
<tr>
<td>type</td>
<td>The type argument identifies the type of object. Types range from 0-17 (see Table 6 on page 162).</td>
</tr>
<tr>
<td>ispolicyattachable</td>
<td>Determines whether you can attach an ACL policy to this object.</td>
</tr>
</tbody>
</table>
Table 6. Object types

<table>
<thead>
<tr>
<th>Object types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – unknown</td>
<td>10 – non-existent object</td>
</tr>
<tr>
<td>1 – secure domain</td>
<td>11 – container object</td>
</tr>
<tr>
<td>2 – file</td>
<td>12 – leaf object</td>
</tr>
<tr>
<td>3 – executable program</td>
<td>13 – port</td>
</tr>
<tr>
<td>4 – directory</td>
<td>14 – application container object</td>
</tr>
<tr>
<td>5 – junction</td>
<td>(required for delegated administration)</td>
</tr>
<tr>
<td>6 – WebSEAL server</td>
<td>15 – application leaf object</td>
</tr>
<tr>
<td>7 – unused</td>
<td>16 – management object</td>
</tr>
<tr>
<td>8 – unused</td>
<td>17 – unused</td>
</tr>
<tr>
<td>9 – HTTP server</td>
<td></td>
</tr>
</tbody>
</table>

For example:

```
pdadmin>object create /Management/Groups/Travel "Travel Container Object"
```
```
14 ispolicyattachable yes
```

For example, you can also use the `pdadmin group create` command to create a group container object. See “Create groups.”

For more information about the `pdadmin object create` command, see the IBM Tivoli Access Manager for e-business Command Reference.

Create groups

To create a new group and optionally place this group in a group container object using the Web Portal Manager or the `pdadmin` command line utility, log in to the desired domain as a domain administrator.

Web Portal Manager

To create a new group and optionally place this group in a group container object using Web Portal Manager:

1. Log in to the domain.
2. Click **Group + Create Group**.
3. Type a **Group Name** for the group (for example, group1). This field is required.
4. Optionally, type a **Description** for the group (for example, Travel group 1).
5. Type a **Registry GID**. This field is required. The registry GID specifies the location in the registry database where the new group is created. For example: `cn=travel,c=us`. Lotus Notes users require the full path name for the user being created. For example: `travel/US`.
6. Optionally, type the path in the **Object Container field** to the Tivoli Access Manager object space where the group is to be created. Be sure to type the full path:

   `/Management/Groups/Travel`

7. Click **Create**. The new group will be displayed as a link. Select the link and the properties for the new group are displayed.
To create a new group and optionally place this group in a group container object using the `pdadmin` utility, log in to the domain and use the `pdadmin group create` command.

For example, to create a new group and optionally place this group in a group container object. If the container object does not currently exist, it is automatically created:

```
pdadmin> group create group_name dn cn [group_container]
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>group_name</code></td>
<td>Name of the new group object.</td>
</tr>
<tr>
<td><code>dn</code></td>
<td>Distinguished name for the new group.</td>
</tr>
<tr>
<td><code>cn</code></td>
<td>Common name for the new group.</td>
</tr>
<tr>
<td><code>group_container</code></td>
<td>Relative path name for the group container object where this new group should be located. If no group container object is specified, the group is placed under /Management/Groups.</td>
</tr>
</tbody>
</table>

For example:

```
pdadmin> group create group1 "cn=travel,c=us" Group1 Travel
```

```
pdadmin> group create group2 "cn=travel,c=us" Group2 Travel
```

**Notes:**

1. All new group container objects that you create appear under the default /Management/Groups container. To create a container at another sublevel, use a relative path name for the `group_container` variable.
2. The `group create` command does not allow you to create a group container object without a group.
3. To add a new group to the object space, the administrator must have create (N) permission on the ACL governing the associated group container object. If no group container object is specified, the administrator ACL entry (with the create permission) must be specified in the ACL governing the /Management/Groups container. At installation, a single default ACL (default-management), which is attached to /Management, defines the permissions on all groups and group containers. You must add appropriate explicit ACLs to customize this control.
4. You can add multiple groups to a single group container.
   The ACL on the group container object controls (through inheritance) all groups located under the container object. The container object and its groups are now the domain of the administrator with the delegated responsibilities.

5. The placement of a new group in the object space is fixed on creation.
   As soon as a group is created, you can move its position only by deleting the group from the object space (but not LDAP) and then importing the group to a new location (users in the group are maintained).

For more information about the `pdadmin group create` command, see the *IBM Tivoli Access Manager for e-business Command Reference*.

**ACL policies affecting group management**

Authorization to control a group of users is obtained by attaching an appropriate ACL to the group object or group container object.

The ACL, constructed and attached by a higher-level administrator, should contain the appropriate permissions for the actions that must be performed by the delegated administrator of that group (or groups).

If the group is located under the `/Management/Groups` section of the object space, the ACL must be attached to `/Management/Groups` or the group itself.

If the group is located under a group container object, the ACL must be attached to the group container object or the group itself. If you attach the ACL to the `/Management/Groups` container object, the ACL would impact all other group container objects located below `/Management/Groups` in the object space.

The ACL that is attached to one of these locations (or inherited from above) determines:

- Who controls the group object and the users in the group
- What actions can be performed on the group and its users

For example, in [Figure 28 on page 163](#) an ACL on `/Management/Groups/Travel` defines permissions to control both group1 and group2.

The following operations and ACL permissions are appropriate for group management:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>create (a new group) import (group data from the user registry)</td>
<td>N (create)</td>
</tr>
<tr>
<td>delete (a group)</td>
<td>d (delete)</td>
</tr>
<tr>
<td>show (group details)</td>
<td>v (view)</td>
</tr>
<tr>
<td>modify (group description)</td>
<td>m (modify)</td>
</tr>
<tr>
<td>add (an existing user to a group)</td>
<td>A (add)</td>
</tr>
<tr>
<td>remove (a user member of the group)</td>
<td>A (add)</td>
</tr>
</tbody>
</table>

You can use the appropriate `pdadmin` utility commands, or the Web Portal Manager, to perform these operations.
Attention

The add (A) permission is powerful because it allows you to add any existing user to your groups. If an outside user is placed into a group, the administrator of that group now has control of that user (and might share control of the user with administrators of other groups where that user is a member). This permission is best granted only to high-level administrators who are responsible for user and group organization and corporate policy.

Because of its power, use caution when assigning an administrator with the A permission. A delegated administrator with the A permission should not have m, W, N, or d permissions.

Notes:

- The create (N) permission must be located in an ACL that is attached to /Management/Groups or on a group container object.
- All other permissions listed can be located in an ACL attached to /Management/Groups, a group container object, or the group object itself.

ACL policies affecting user management

The group administrator can perform an action on a user if the administrator has the appropriate permission defined on any of the groups where that user is a member.

The following operations and ACL permissions are appropriate for user management:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>create (a new user within one or more specified groups) import (user data from the user registry)</td>
<td>N (create)</td>
</tr>
<tr>
<td>delete (a user)</td>
<td>d (delete)</td>
</tr>
<tr>
<td>show (user details)</td>
<td>v (view)</td>
</tr>
<tr>
<td>modify (user description)</td>
<td>m (modify)</td>
</tr>
<tr>
<td>account valid</td>
<td>m (modify)</td>
</tr>
<tr>
<td>reset password</td>
<td>W (password)</td>
</tr>
<tr>
<td>password-valid</td>
<td>W (password)</td>
</tr>
</tbody>
</table>

You can use the appropriate pdadmin utility commands, or the Web Portal Manager, to perform these operations.

Notes:

- The create (N) permission (in the group ACL or group container ACL) allows you to create or import a user and enter that user into the groups you control.

  user create user1 "cn=user1,c=us" user1 user1 adcdc group1
  user import user2 "cn=user2,c=us" group1

- You can also create a user without designating a group. In this case, however, the create (N) permission must be located in an ACL on the /Management/Users container object.

  The ACL attached to /Management/Users defines the permissions for all users (whether they are members of a group or not).
• A group administrator can perform an operation on a user if that administrator has the appropriate permission defined in any group where that user is a member.

• If a user is not a member of any group, an administrator must have appropriate permissions in an ACL on /Management/Users to perform operations on that user.

• The password (W) permission is appropriate for help desk operators who must assist users who have lost their passwords.

   The operator can reset the lost password to some known value, and then set user modify password-valid (pdadmin) to no. This action would force the user to change the password at the next login. Note that setting user modify password-valid to no for a user does not indicate if the password is not valid due to the max-password-age policy, which is a global setting. The policy set max-password-age command sets the maximum time before a password expires.

• The view (v) permission is used to control the output of user list, user list-dn, user show groups, group list, and group list-dn commands. The view permission is used to filter the output of these commands. If the user does not have view permission on a group or user that is being returned by the command, that group or user is filtered from the output.

---

**Delegated administration security policy**

The previous sections described separately how to delegate administration of security policy for protecting resources in your domain and also how to delegate management of the users who access those resources. These two individual aspects of delegated administration often need to be combined to establish a complete delegated administration security policy.

Great care, however, must be taken when doing this. In particular, you must be careful which permissions you grant in combination with each other.

For example, the A permission should never be granted together with the m, W, or d permissions except to the most powerful and trusted administrators (and maybe not at all). The consequence of granting both A and W to administrators is that the administrators can add any user to the group for which they have these permissions and then change that user’s password. Any user can be chosen, including a more senior administrator or even sec_master. In this way, a malicious administrator could gain full access to the system by logging on as that senior user.

The consequence of granting the A and m permissions together are similar except that an administrator with both of these permissions needs only this combination to disable any account in the group. The consequence of granting the A and d permissions together are similar except that an administrator with both of these permissions needs only this combination to delete any user ID in the group.

When defining a complete delegated administration policy, these constraints imply a certain structure and use to your user groups.

You must establish groups that you use to delegate user management tasks—such as creating new users, deleting users and resetting users’ passwords. Administrators that perform user administration tasks should have the N, d, m, W, and v permissions to create, delete, modify (disable or change description), reset or invalidate passwords, and view users they are responsible for managing. These groups are used only for delegating user management. These groups should not be used for protecting other resources in the domain.
You must also establish groups that you use to delegate management of a security policy for protected resources within the domain. Administrators controlling security policy for these groups should have the A and v permissions but none of the N, d, m, or W permissions. These groups are used to control access to the real resources that need protecting.

Example:

Suppose that you have a Web space accessible to the Internet with resources that should be:

- Publicly accessible
- Accessible only to customers and employees
- Accessible only to employees

The space can be structured as follows:

```
/WebSEAL/
    www.company.ibm.com/
        customers/
        sales/
```

An ACL at the root of the www.company.ibm.com Web space allows public access to everything in the Web space. An ACL at customers allows access to customers and sales people. Another ACL at sales allows access only to sales people. These ACLs might look like the following example:

```
public-access
    user sec_master  abcTdm1rx
    any-other        Tlrx
    unauthenticated  Tlrx

customer-access
    user sec_master  abcTdm1rx
    group customers  Tlrx
    group sales      Tlrx
    any-other        Tlrx
    unauthenticated  Tlrx

sales-access
    user sec_master  abcTdm1rx
    group sales      Tlrx
    any-other        Tlrx
    unauthenticated  Tlrx
```

These ACLs would be attached respectively to the following:

```
/WebSEAL/www.company.ibm.com
/WebSEAL/www.company.ibm.com/customers
/WebSEAL/www.company.ibm.com/sales
```

Suppose that you have the following delegated user administration policy. Sales people (members of the sales group) are allowed to create new accounts for customers and grant them access to the customers portion of the Web space. Only administrators (members of the sales-admin group) are allowed to manage accounts for new sales people.

The following group structure implements this policy:

```
/Management/
Groups/
    sales            <- ACL sales-admin
    sales-users      <- ACL sales-users-admin
    customers        <- ACL customers-admin
    customers-users  <- ACL customers-users-admin
```
The **sales-admin** ACL is used to administer membership of the sales group, which in turn, is used to control access to the sales-people-only portion of the Web space. The only permission required is for the **sales-admin** group to be able to add and remove users from this group. The view (v) permission is also useful to administrators to allow them to view the group membership and the users in the group.

```
sales-admin
    group super-admin Tabc
    group admin TAv
```

The **sales-users-admin** ACL, by attachment to the **sales-users** group, controls who can manage users who are members of the **sales-users** group (this is the **sales-admin** group again).

```
sales-users-admin
    group super-admin Tabc
    group admin TNWdmv
```

Similarly, the **customers-admin** ACL is used to administer membership of the **customers** group, which in turn, is used to control access to the customers-only portion of the Web space.

```
customers-admin
    group super-admin Tabc
    group admin sales TAv
```

The **customers-users-admin** ACL, by attachment to the **customers-users** group, controls who can manage the members of the **customers-users** group (this the sales group again). We also allow members of the **sales-admin** group to manage customers.

```
customers-users-admin
    group super-admin Tabc
    group sales TNWdmv
    group admin TNWdmv
```

Notice in each ACL, a **super-admin** group entry is granted attach, browse, and control permission. Members of the **super-admin** group are responsible for administering these ACLs.
Chapter 17. Log and routing files

The contents of log files and can be useful sources of information when monitoring and troubleshooting the activity of Tivoli Access Manager servers. Log files can capture any error messages and any warning messages that are generated by Tivoli Access Manager servers. The directory location for serviceability message log files can be different, depending on whether Tivoli Common Directory is configured.

The routing files control the location and configuration of the serviceability message log files. This chapter describes the configuration syntax used in the routing files and defines the default file name and location of the message log files.

The IBM Tivoli Access Manager for e-business Problem Determination Guide describes how to turn on tracing and logging, including Tivoli Common Directory logging.

This chapter contains the following sections:

- “Base routing files”
- “Common log file directories” on page 174
- “XML log viewer” on page 176
- “Log file and routing file tasks” on page 176

Base routing files

A routing file is an ASCII file that contains commands that control the configuration of messages. Each Tivoli Access Manager server has its own routing file.

pdmgrd_routing for the Tivoli Access Manager policy server (pdmgrd)
pdacld_routing for the Tivoli Access Manager authorization server (pdaclid)
pdmgrproxyd_routing for the Tivoli Access Manager policy proxy server (pdmgrproxyd)
routing for Tivoli Access Manager serviceability routing

Serviceability message logs

Tivoli Access Manager Base serviceability messages are controlled by a Tivoli Access Manager Base routing file. The routing files are ASCII files that contain additional information in the form of comment lines. Entries in the server’s routing file determine the types of serviceability messages that are logged.

You can disable or enable any type of message logging by adding or removing, respectively, the comment character (#) at the beginning of the line in the routing file.
### Table 7. UNIX and Windows routing files

<table>
<thead>
<tr>
<th>Server</th>
<th>UNIX:</th>
<th>Windows:</th>
</tr>
</thead>
<tbody>
<tr>
<td>pdmgrd</td>
<td>FATAL:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdmgrd_utf8.log</td>
<td>FATAL:STDOUT:-;UTF8FILE:%PDDIR%/log/msg__pdmgrd_utf8.log</td>
</tr>
<tr>
<td></td>
<td>ERROR:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdmgrd_utf8.log</td>
<td>ERROR:STDOUT:-;FILE:%PDDIR%/log/msg__pdmgrd_utf8.log</td>
</tr>
<tr>
<td></td>
<td>WARNING:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdmgrd_utf8.log</td>
<td>WARNING:STDOUT:-;FILE:%PDDIR%/log/msg__pdmgrd_utf8.log</td>
</tr>
<tr>
<td></td>
<td>NOTICE:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdmgrd_utf8.log</td>
<td>NOTICE:FILE:%PDDIR%/log/msg__pdmgrd_utf8.log</td>
</tr>
<tr>
<td></td>
<td>#NOTICE_VERBOSE:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdmgrd_utf8.log</td>
<td>#NOTICE_VERBOSE:STDOUT:-;UTF8FILE:%PDDIR%/log/msg__pdmgrd_utf8.log</td>
</tr>
<tr>
<td></td>
<td>Note: On a Windows system, the special environment variable PDDIR is set at runtime to the Tivoli Access Manager installation directory.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>pdacld</th>
<th>UNIX:</th>
<th>Windows:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FATAL:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdacld_utf8.log</td>
<td>FATAL:STDOUT:-;UTF8FILE:%PDDIR%/log/msg__pdacld_utf8.log</td>
</tr>
<tr>
<td></td>
<td>ERROR:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdacld_utf8.log</td>
<td>ERROR:STDOUT:-;FILE:%PDDIR%/log/msg__pdacld_utf8.log</td>
</tr>
<tr>
<td></td>
<td>WARNING:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdacld_utf8.log</td>
<td>WARNING:STDOUT:-;FILE:%PDDIR%/log/msg__pdacld_utf8.log</td>
</tr>
<tr>
<td></td>
<td>NOTICE:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdacld_utf8.log</td>
<td>NOTICE:FILE:%PDDIR%/log/msg__pdacld_utf8.log</td>
</tr>
<tr>
<td></td>
<td>#NOTICE_VERBOSE:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdacld_utf8.log</td>
<td>#NOTICE_VERBOSE:STDOUT:-;UTF8FILE:%PDDIR%/log/msg__pdacld_utf8.log</td>
</tr>
<tr>
<td></td>
<td>Note: On a Windows system, the special environment variable PDDIR is set at runtime to the Tivoli Access Manager installation directory.</td>
<td></td>
</tr>
</tbody>
</table>
Table 7. UNIX and Windows routing files (continued)

<table>
<thead>
<tr>
<th>pdmgrproxyd server</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UNIX:</strong></td>
</tr>
<tr>
<td>FATAL:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdmgrproxyd_utf8.log:644:ivmgr:ivmgr</td>
</tr>
<tr>
<td>ERROR:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdmgrproxyd_utf8.log:644:ivmgr:ivmgr</td>
</tr>
<tr>
<td>WARNING:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdmgrproxyd_utf8.log:644:ivmgr:ivmgr</td>
</tr>
<tr>
<td>NOTICE:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdmgrproxyd_utf8.log:644:ivmgr:ivmgr</td>
</tr>
<tr>
<td>#NOTICE_VERBOSE:STDOUT:-;UTF8FILE:/var/PolicyDirector/log/msg__pdmgrproxyd_utf8.log:644:ivmgr:ivmgr</td>
</tr>
<tr>
<td><strong>Windows:</strong></td>
</tr>
<tr>
<td>FATAL:STDOUT:-;UTF8FILE:%PDDIR%\log\msg__pdmgrproxyd_utf8.log</td>
</tr>
<tr>
<td>ERROR:STDOUT:-;UTF8FILE:%PDDIR%\log\msg__pdmgrproxyd_utf8.log</td>
</tr>
<tr>
<td>WARNING:STDOUT:-;UTF8FILE:%PDDIR%\log\msg__pdmgrproxyd_utf8.log</td>
</tr>
<tr>
<td>NOTICE:STDOUT:-;UTF8FILE:%PDDIR%\log\msg__pdmgrproxyd_utf8.log</td>
</tr>
<tr>
<td>#NOTICE_VERBOSE:STDOUT:-;UTF8FILE:%PDDIR%\log\msg__pdmgrproxyd_utf8.log</td>
</tr>
</tbody>
</table>

**Note:** On a Windows system, the special environment variable **PDDIR** is set at runtime to the Tivoli Access Manager installation directory.

The server configuration files pick up the information from the routing files. However, if for any reason the routing files have been deleted, the log-file stanza entry for the appropriate server will be used instead. The default configuration location is one of the following:

**UNIX:** /var/PolicyDirector/log/  
**Windows:** pd_dir\log\

For example, the configured log-file stanza entry in the authentication server configuration file ivacld.conf would be similar to the following:

**UNIX:**
```
[ivacld]
logfile = /var/PolicyDirector/log/msg__pdacld_utf8.log
```

**Windows:**
```
[ivacld]
log-file = pd_install_dir\log\msg__pdacld_utf8.log
```

**Routing file output syntax**

By default, when Tivoli Access Manager Base runs in the foreground, messages are handled in one of the following manners:

- Messages are sent to the screen as STDOUT or STDERR.
- Messages are sent to the appropriate routing log file entries in the log directory.
By default, when Tivoli Access Manager Base runs in the background, messages are sent to log files specified in the server routing files. In this example of the routing file template default entries, the syntax controls log output. The syntax format uses a colon (:) as a separator:
FATAL:STDOUT:-;FILE:/var/PolicyDirector/log/msg__fatal_utf8.log:666:ivmgr:ivmgr

Where:

message_type
The types of serviceability log messages include:

- Fatal
  Fatal error exit. An unrecoverable error (such as database corruption) has occurred that might require manual intervention for the correction. The program usually ends (terminates) immediately after such an error.
- Error
  Error detected. An unexpected event, which is not terminal (such as a timeout) or is correctable by human intervention, has occurred. The program continues operation although some functions or services might no longer be available. This severity level also indicates that a particular request or action could not be completed.
- Warning
  Correctable error. An error occurred that was automatically corrected (for example, a configuration file was not found, and default values were used instead). This severity level is also used to indicate a condition that might be an error if the effects are undesirable or to indicate a condition which, if not corrected, will eventually result in an error.
- Notice
  Informational notice: A significant routine major event has occurred. For example, a server has started.
- Notice_verbose
  Verbose information notice. A significant routine event has occurred. For example, a directory entry has been removed.

foreground_output
When Tivoli Access Manager Base runs in the foreground, the log messages can be sent to the screen as STDOUT or STDERR.

background_output
When Tivoli Access Manager Base runs in the background, specifies how the messages of a given severity level should be processed, and how the messages are to be output to these types of log files:

FILE or TEXTFILE filename
The default FILE or TEXTFILE entry for the configured log files that specifies log files are not to be recycled or rolled over. These options let you specify that the log output is to be directed to the file name that immediately follows. Another parameter determines whether the log file grows forever or whether the log file can span a limited number of generations.

UTF8FILE
A UTF8FILE output entry is equivalent to the TEXTFILE entry except the data is encoded in UTF-8 format.
**XMLSTDERR**
The entry that is equivalent to STDERR except that the data will be written in Log XML format and encoded in UTF-8.

**XMLSTDOUT**
The entry that is equivalent to STDOUT except that the data will be written in Log XML format and encoded in UTF-8.

**XMLFILE**
The entry that is equivalent to TEXTFILE except that the data will be written in Log XML format and encoded in UTF-8.

**DISCARD**
This entry specifies that you do not want to record messages of a specified message type (severity level).

These files are configured to grow sequentially without limit. When configured to grow without limit, you must watch the disk space. When you run out of disk space, you must delete existing log files or prune the log files periodically.

**routing_path**
The default directories for server routing files are:

UNIX: /opt/PolicyDirector/etc/
Windows: C:\Program Files\Tivoli\Policy Director\etc\

This chapter uses the `routing_path` variable as the convention for the routing file default path.

**log_filename**
The default message log file template uses these log file names for log files in UTF–8 format:
```
msg__fatal_utf8.log
msg__error_utf8.log
msg__warning_utf8.log
msg__notice_utf8.log
msg__verbose_utf8.log
```

Note that there are 2 underscore characters following `msg` in the message log file name.

The Tivoli Access Manager server message log files use different log file names that more closely relate to the names of the servers. See the log file names noted in Table 7 on page 170.

**perm_filename**
The permission extensions on the file (for example, 666). The permissions extensions are only valid on UNIX.

**user_name**
The user that owns the file (for example, ivmgr). This option applies only to UNIX.

**group_name**
The group that owns the file (for example, ivmgr). This option applies only to UNIX.

If the FILE entry for the configured log files specifies `n` and `m` values, such as:
```
NOTICE:FILE.10.1000:/var/PolicyDirector/log/msg__syslog.log
```
The \( n \) represents the number of files and the \( m \) represents the number of appends allowed in each file. For example, FILE.10.1000 specifies that there are to be a maximum of 10 message log files, with each file containing no more than 1,000 log entries in each file.

The message log files wrap around to the first file after the last file has reached its limit or when the server is stopped and restarted. When a message log file is reused, the existing records are written over (erased).

```
msg_notice_utf8.log.1
msg_notice_utf8.log.2
.
.
msg_notice_utf8.log.10
```

### Common log file directories

Tivoli Access Manager provides Tivoli Common Directory logging capabilities. When enabled, the common Tivoli Common Directory logging directory is where all Tivoli Access Manager serviceability message log files are located. Other types of application log files continue to be located in their installation locations.

Tivoli products and applications that provide Tivoli Common Directory support store their serviceability-related files (such as logs, first-failure data capture (FFDC) data, and serviceability scripts) in the same central location. Locating all Tivoli product log files in one location makes it easier to find and investigate not only the Tivoli Access Manager log files when problem determination is needed, but also to find and investigate the log files of other Tivoli products.

### Tivoli Common Directory default location

The following common directory structure centralizes log files, first failure data capture files, and serviceability scripts for all Tivoli products. The directory structure used includes:

```
tivoli_common_dir/XXX/subdir/[qualifier]
```

where:

- **tivoli_common_dir** Represents the central location for Tivoli Common Directory files. You are asked during configuration to provide a Tivoli Common Directory location only if you have never used Tivoli Common Directory for any Tivoli product.
- **XXX** Represents the 3-letter identifier to use for the product’s message log files.
- **subdir** Represents the subdirectory that divides the message log files into categories. The subdirectories include:
  - logs for log files
  - ffdc for first failure data capture files
  - scripts for serviceability scripts
- **date** Represents a date-dependent subdirectory (in format yyyymmdd) for FFDC data captured on a given date.
- **[qualifier]** Represents optional subdirectories for specific named instances or components of the product. The qualifier might be needed to avoid
collisions whenever there are multiple components or instances attempting to store different files using the same fully qualified path.

The default installation location for Tivoli Common Directory (tivoli_common_dir) is:

UNIX: /var/ibm/tivoli/common
Windows: C:\Program Files\ibm\tivoli\common\

On UNIX, the Tivoli Common Directory should have the 771 permission and be owned by group tivoli.

Note: After you define another Tivoli Common Directory location or accept the default location, you cannot change the location.

Tivoli Common Directory properties file

The current location of the Tivoli Common Directory is specified in the log.properties file. The default path to this configuration file is:

UNIX: /etc/ibm/tivoli/common/cfg/log.properties
Windows: C:\Program Files\ibm\tivoli\common\cfg\log.properties

On UNIX, the file should have the 664 permission and should be owned by group tivoli.

Tivoli Access Manager default location

During Tivoli Access Manager C runtime or Java runtime configuration, if the Tivoli Common Directory location has not yet been specified, you are asked to define the Tivoli Common Directory location (TCD) for your Tivoli Access Manager serviceability message and trace log files.

Tivoli Access Manager defines a Tivoli Common Directory default log file location, and uses the logs subdirectory for storing the message and trace logs. The default location will be displayed, and you can change it if you have never installed Tivoli Common Directory before.

The default installation location syntax is:

TCD/XXX/logs/

where:

TCD Represents the tivoli_common_dir directory extension specified by Tivoli Access Manager, if the Tivoli Common Directory has not yet been defined by the installation of another Tivoli product.

XXX Represents the 3-letter identifier to use for the product’s message log files. For Tivoli Access Manager, the identifiers include:

HPD for IBM Tivoli Access Manager Base
HPW for IBM Tivoli Access Manager WebSEAL
AMZ for IBM Tivoli Access Manager Web Security plug-ins
AWX for IBM Tivoli Access Manager WebSphere Application Server
AWL for IBM Tivoli Access Manager BEA WebLogic Server
AWD for Tivoli Access Manager Edge Server
AOS for IBM Tivoli Access Manager for Operating Systems
DRQ for IBM Tivoli Access Manager for Business Integration
The subdirectory used for Tivoli Access Manager. Note that only one subdirectory (logs) for the message and trace log files is defined.

If Tivoli Common Directory is configured, the path of the common logging directory is placed in the tivoli_common_dir stanza entry in the pd.conf configuration file. For example:

[pdrt]
tivoli_common_dir = C:\Program Files\Policy Director\TAMBase\ 

If configured for Tivoli Common Directory, the configured log-file stanza entry in the server configuration file will reflect the Tivoli Common Directory installation location. For example, if Tivoli Common Directory is enabled for the authorization server (pdacld), the log_file stanza entry would be similar to the following:

UNIX using the Tivoli Common Directory location:

[ivacld]
log-file = /PolicyDirector/TAMBase/HPD/logs/msg__pdacld_utf8.log

Windows using the Tivoli Common Directory location:

[ivacld]
log-file = C:\Program Files\Policy Director\TAMBase\HPD\logs\msg__pdacld_utf8.log

XML log viewer

To more easily view XML output, use the Tivoli XML Log Viewer tool that is provided with Tivoli Access Manager. The XMLFILE, XMLSTDERR, and XMLSTDOUT keywords within the routing file discussed in the "Routing file output syntax" on page 171 are used to produce XML message logs and XML trace logs. See the IBM Tivoli Access Manager for e-business Problem Determination Guide to learn more about viewing XML format and to examine step-by-step the message logging and trace logging files XML output.

Log file and routing file tasks

This section describes log file and routing file tasks available to the administrator.

- Change the message log files location
- "Log messages in log XML format" on page 177
- "Enable viewing of trace records" on page 178

Change the message log files location

To change the directory for the Tivoli Access Manager server-specific message log files:

1. Go to the directory where the routing files are located. The default directory location is one of the following:

   UNIX : /opt/PolicyDirector/etc/
   Windows : C:\Program Files\Tivoli\Policy Director\etc\ 

2. Select one of the appropriate server-related routing file to edit:
   - pdmgrd_routing for the Tivoli Access Manager policy server (pdmgrd)
   - pdacld_routing for the Tivoli Access Manager authorization server (pdacld)
   - pdmgrproxyd_routing for the Tivoli Access Manager policy proxy server (pdmgrproxyd)
   - routing for Tivoli Access Manager general serviceability information
3. Edit the file and locate the section titled Sequential Logging.

4. Change the default location for the message log files, as appropriate.

In the following policy server (pdmgrd) example, you can change from the default routing_path installation location of /var/PolicyDirector/log/:

```
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
```

to a different directory location of /myTAMlogs/:

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

5. Exit and save the routing file.

Remember to prune log files periodically to prevent them from becoming too large.

Log messages in log XML format

To log messages in XML format:

1. Go to the directory where the routing files are located. The default directory location is one of the following:

   **UNIX**: /opt/PolicyDirector/etc/
   **Windows**: C:\Program Files\Tivoli\Policy Director\etc\n
2. Select one of the appropriate server-related routing file to edit:
   - pdmgrd_routing for the Tivoli Access Manager policy server (pdmgrd)
   - pdacld_routing for the Tivoli Access Manager authorization server (pdacld)
   - pdmgrproxyd_routing for the Tivoli Access Manager policy proxy server (pdmgrproxyd)
   - routing for Tivoli Access Manager general serviceability information

3. Find a line similar to the following in the routing file:

   ```
   ERROR:STDOUT:-;XMLFILE:%PDDIR%/log/msg__error.log
   ```

   For example, to change the line to specify that ERROR messages should be logged in XML format, instead of text format, to both STDOUT and to the file msg__error.log:

   ```
   ERROR:XMLSTDOUT:-;XMLFILE:%PDDIR%/log/msg__error.log
   ```

   where %PDDIR% is the Tivoli Access Manager UNIX directory variable.

   See "Routing file output syntax" on page 171 for more details on how to specify additional changes.
Enable viewing of trace records

The name of each event category is written to a trace event record as it is instantiated.

To enable tracing during startup and be able to view trace records:

1. Edit the appropriate routing file for the server. Routing file names include:
   - pdmgrd_routing for the Tivoli Access Manager policy server (pdmgrd)
   - pdmgrd_routing for the Tivoli Access Manager authorization server (pdacld)
   - pdmgrproxyd_routing for the Tivoli Access Manager policy proxy server (pdmgrproxyd)

2. Add a line similar to the following to the routing file:
   
   *:*:TEXTFILE:%PDDIR%/log/trace_%ld.log

   Or, remove the pound sign (#) at the beginning of this line, if it already exists in the routing file, to allow viewing of trace records.

3. Change this line, if you want to log this debug trace data in XML log format. For example, you can change the line to send the output to an XML file instead of a text file:

   *:*:XMLFILE.10.1000:%PDDIR%/log/trace__%ld.log;XMLSTDERR:-

   where %PDDIR% is the Tivoli Access Manager UNIX directory variable. The XMLFILE.10.1000 syntax is explained in “Routing file output syntax” on page 171.

See the IBM Tivoli Access Manager for e-business Problem Determination Guide where the trace records are discussed in greater detail.
Chapter 18. XML output for logging and auditing logs

Audit events are captured in the audit trail in a standard format using the Extensible Markup Language (XML) tags. XML is only an intermediary step to delivering a presentation view of the data. The XML file is in ASCII format and can be read directly or passed to other external parsing engines for further analysis.

**DTD intermediate format**

As an audit administrator, you are expected to select and extract events according to your own criteria. This activity might include reformatting each event by applying an appropriate Document Type Definition (DTD) or schema for the analysis tool that you are using. The DTD is an intermediate format that provides a description of the data that can be captured.

**Data blocks and output fields**

An entire audit trail does not represent a single XML document. Each audit event within the file is written as an isolated XML data block. Each data block conforms to the rules of standard XML syntax.

For example, the following data block is an audit record for getting user authorization credentials:

```xml
<event rev="1.2">
  <date>2003-11-14-16:25:08.341+00:00</date>
  <outcome status="0">0</outcome>
  <originator blade="pdmgrd"><component rev="1.2">azn</component>
    <action>0</action>
    <location>phaedrus</location>
  </originator>
  <accessor name="">
    <principal auth="IV_LDAP_V3.0">sec_master</principal>
  </accessor>
  <target resource="3"><object>IV_LDAP_V3.0:sec_master</object></target>
  <data>azn_id_get_creds</data>
</event>
```

Table[Table 8 on page 180](180) describes the XML output fields that are possible by using the default Tivoli Access Manager DTD elements. If you create your own DTD, each field must represent the events that you selected and extracted according to your own criteria.
Table 8. Names and descriptions for XML output fields

<table>
<thead>
<tr>
<th>Output Field Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| <event> ... </event> | Auditing event. Each auditing event captures the result of an action that a principal attempts on a target object. The event field can contain these fields: - date - outcome - accessor - target - data Because Tivoli Access Manager auditing uses a standard record format, not all fields are relevant to every event recorded. Fields that are not relevant for a particular event might contain a default value. Attributes for this field can include: - rev - link Example:  
```
<event rev="1.2">
  <date>2003-11-14-16:25:08.341+00:00I-----</date>
  <outcome status="0">0</outcome>  
  ...
  <accessor name="">
    <principal auth="IV_LDAP_V3.0">sec_master</principal>
  </accessor>
  <target resource="3">
    <object>IV_LDAP_V3.0:sec_master</object>
  </target>
  <data>azn_id_get_creds</data>
</event>
```
| <date> ... </date> | Current date and timestamp. The format for the date and timestamp is:  
```
yyyy-mm-dd-hh:mm:ss.000+00:00I-----
```
Where:  
- **yyyy-mm-dd**  
  - Relates to the year (yyyy), the month (mm), and the day (dd).  
- **hh:mm:ss**  
  - Relates to hours (hh), minutes (mm), and seconds (ss).  
- **000+00:**  
  - Refers to the time zone.  
- **00I-----**  
  - Refers to time inaccuracy in milliseconds. This is used for legacy purposes only. Example:  
```
<event rev="1.2">
  <date>2003-11-14-16:25:08.341+00:00I-----</date>
  ...
</event>
```
Table 8. Names and descriptions for XML output fields (continued)

<table>
<thead>
<tr>
<th>Output Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;outcome&gt;</code> ...</td>
<td><strong>Outcome</strong> of the event. The possible outcome values include:</td>
</tr>
<tr>
<td></td>
<td>0  Success</td>
</tr>
<tr>
<td></td>
<td>1  Failure</td>
</tr>
<tr>
<td></td>
<td>2  Pending</td>
</tr>
<tr>
<td></td>
<td>3  Unknown</td>
</tr>
<tr>
<td></td>
<td>Information about the outcome, along with the action, the principal’s credentials, and the target object are captured in a common format header of the audit record.</td>
</tr>
<tr>
<td></td>
<td>Attributes for this field always includes:</td>
</tr>
<tr>
<td></td>
<td>• status</td>
</tr>
<tr>
<td></td>
<td>Example of a failed event:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;outcome status=&quot;412668954&quot;&gt;1&lt;/outcome&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Use the <code>pdadmin errtext</code> command to provide interpretation for the status code (412668954) of a failed event.</td>
</tr>
<tr>
<td></td>
<td>If the error is not identified by the <code>pdadmin errtext</code> command, then the status code did not originate in the IBM Tivoli Access Manager for e-business base product. See your application’s problem determination guide for additional status code definitions.</td>
</tr>
<tr>
<td></td>
<td>Example of a successful event:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;event rev=&quot;1.2&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td><code>&lt;outcome status=&quot;0&quot;&gt;0&lt;/outcome&gt;</code></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td><code>&lt;/event&gt;</code></td>
</tr>
<tr>
<td><code>&lt;originator&gt;</code> ...</td>
<td><strong>Server</strong> that originated the event being logged.</td>
</tr>
<tr>
<td></td>
<td>The originator field can contain these fields:</td>
</tr>
<tr>
<td></td>
<td>• component</td>
</tr>
<tr>
<td></td>
<td>• action</td>
</tr>
<tr>
<td></td>
<td>• location</td>
</tr>
<tr>
<td></td>
<td>Valid attributes include:</td>
</tr>
<tr>
<td></td>
<td>• blade</td>
</tr>
<tr>
<td></td>
<td>The blade represents the server that originated the event. For example, <code>pdmgrd</code> is the Tivoli Access Manager policy server.</td>
</tr>
<tr>
<td></td>
<td>Examples:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;event rev=&quot;1.2&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td><code>&lt;originator blade=&quot;pdmgrd&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;component rev=&quot;1.2&quot;&gt;azn&lt;/component&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;action&gt;0&lt;/action&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;location&gt;phaedrus&lt;/location&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/originator&gt;</code></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td><code>&lt;/event&gt;</code></td>
</tr>
</tbody>
</table>
### Table 8. Names and descriptions for XML output fields (continued)

<table>
<thead>
<tr>
<th>Output Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;component&gt;</code></td>
<td>Audit events, categorized by the server functionality that generates them. Some functionality is common across Tivoli Access Manager servers while other functionality is server-specific. Valuable values include: <code>azn</code> Captures authorization events. <code>authn</code> Captures authentication events. <code>mgmt</code> Captures management events. <code>http</code> Captures WebSEAL HTTP events. Refer to the <em>IBM Tivoli Access Manager for e-business WebSEAL Administration Guide</em> for more information about this value. Valid attributes are: • <code>rev</code> Example: <code>&lt;event rev=&quot;1.2&quot;&gt; ... &lt;originator blade=&quot;pdmgrd&quot;&gt; &lt;component rev=&quot;1.2&quot;&gt;azn&lt;/component&gt; &lt;action&gt;0&lt;/action&gt; &lt;location&gt;phaedrus&lt;/location&gt; &lt;/originator&gt; ... &lt;/event&gt;</code></td>
</tr>
<tr>
<td><code>&lt;action&gt;</code></td>
<td>Audit record action code. There are two groups of events: • <em>For authentication or authorization events</em> Audit records for authentication or authorization events contain an action code that distinguishes between: 0 Authentication or authorization events 1 Change password events 2 WebSEAL events • <em>For management events</em> Audit records for management events contain an action code that identifies the Tivoli Access Manager <code>padmin</code> management command. For example, the <code>&lt;action&gt;13702&lt;/action&gt;</code> action code relates to the <code>POP MODIFY</code> action for the <code>padmin pop modify</code> management command. See Table 9 on page 191 which relates the action code reference number for each management command. Information about the action, along with the principal’s credentials, the target object, and the outcome are captured in a common format header of the audit record. Examples: <code>&lt;event rev=&quot;1.2&quot;&gt; ... &lt;originator blade=&quot;pdmgrd&quot;&gt; &lt;component rev=&quot;1.2&quot;&gt;azn&lt;/component&gt; &lt;action&gt;0&lt;/action&gt; &lt;location&gt;phaedrus&lt;/location&gt; &lt;/originator&gt; ... &lt;/event&gt;</code></td>
</tr>
<tr>
<td>Output Field Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| `<location>` ... `<location>` | Location (host name) of the machine. If there is no host name specified, a notation of location not specified will be substituted in the location field.  
Example:  
```xml
<event rev="1.2"> ...
<originator blade="pdmgrd"> ...
<component rev="1.2">azn</component>
&action>0</action> ...
<location>phaedrus</location> ...
<originator> ...
</event>
```

| `<accessor>` ... `<accessor>` | Name of the user causing the event. If there is no user name specified, a notation of name="user not specified" or name="" will be substituted in the accessor field.  
The accessor field can contain this field:  
• principal  
Valid attributes for the accessor field include:  
• name  
Example:  
```xml
<event> ...
<accessor name=""> ...
<principal auth="IV_LDAP_V3.0">sec_master</principal> ...
</accessor>
</event>
```

| `<principal>` ... `<principal>` | User authorization credentials. Generally each event captures the result of an action that a user (principal) attempts on a target object.  
If there is no user name specified, a notation of auth="invalid" will be substituted in the principal field.  
Valid attributes for the principal field include:  
• auth  
• domain  
Information about the principal’s credentials, along with the action, the target object, and the outcome are captured in a common format header of the audit record.  
Examples:  
```xml
<accessor name="">
<principal auth="IV_LDAP_V3.0" domain="Default">testuser2</principal>
</accessor>
```
<table>
<thead>
<tr>
<th>Output Field Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| `<target>`        | Target information. The target field can contain these fields:  
                      - object  
                      - process  
                      - azn  
                      Valid attributes for the target field include:  
                      - resource  
                      The resource attribute of the target field represents a broad categorization of the target object:  
                      0 = AUTHORIZATION  
                      1 = PROCESS  
                      2 = TCB  
                      3 = CREDENTIAL  
                      5 = GENERAL  
                      6 = APPLICATION  
                      7 = AUTHENTICATION  
                      Examples:  
                      `<event>`  
                      ...  
                      `<target resource="7">`<object></object></target>  
                      `<target resource="3">`  
                      `<object>IV_LDAP_V3.0:sec_master</object>`  
                      ...  
                      </target>  
                      ...  
                      </event> |
| `<object>`        | Target object. Authorization audit records can be captured when a target object in the Tivoli Access Manager authorization policy database (protected object space) has a POP attached to it that enables audit functionality. For example:  
                      `<object>/Management</object>`  
                      Information about the target object, along with the action, the principal’s credentials, and the outcome are captured in a common format header of the audit record.  
                      Examples:  
                      `<target resource="3">`  
                      `<object>IV_LDAP_V3.0:sec_master</object>`  
                      ...  
                      </target> |
Table 8. Names and descriptions for XML output fields (continued)

<table>
<thead>
<tr>
<th>Output Field Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| `<azn>`           | Authorization service information. The authorization service checks the access permissions on the target requested object and compares them with the capabilities of the requesting user. The `azn` field can contain these fields:  
  - `perm`  
  - `result`  
  - `qualifier`  
  `<target resource="3">  
  `<azn>  
  `<perm>`64`/perm>`  
  `<result>`0`/result>`  
  `<qualifier>`0`/qualifier>`  
  `</azn>`  
  `</target>` |
| `<perm>`           | Set of controls (permissions) that specifies the conditions necessary to perform certain operations on that resource. The permission can be specified in this field using either the binary number such as `<perm>`64`/perm>` or the letters for the specified action permissions such as `<perm>`Tr`/perm>`.  
  Example:  
  `<target resource="3">  
  `<azn>  
  `<perm>`64`/perm>`  
  `<result>`0`/result>`  
  `<qualifier>`0`/qualifier>`  
  `</azn>`  
  `</target>` |
| `<result>`         | Results of the authorization service check.  
  Example:  
  `<target resource="3">  
  `<azn>  
  `<perm>`64`/perm>`  
  `<result>`0`/result>`  
  `<qualifier>`0`/qualifier>`  
  `</azn>`  
  `</target>` |
| `<qualifier>`      | Qualifier information.  
  Example:  
  `<target resource="3">  
  `<azn>  
  `<perm>`64`/perm>`  
  `<result>`0`/result>`  
  `<qualifier>`0`/qualifier>`  
  `</azn>`  
  `</target>` |
Table 8. Names and descriptions for XML output fields (continued)

<table>
<thead>
<tr>
<th>Output Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;process&gt;</code></td>
<td>Type of process. The process field can include these fields:</td>
</tr>
<tr>
<td>...</td>
<td>• pid (Process ID)</td>
</tr>
<tr>
<td><code>&lt;process&gt;</code></td>
<td>• uid (User ID)</td>
</tr>
<tr>
<td><code>&lt;process&gt;</code></td>
<td>• eid (Effective User ID)</td>
</tr>
<tr>
<td><code>&lt;process&gt;</code></td>
<td>• gid (Group ID)</td>
</tr>
<tr>
<td><code>&lt;process&gt;</code></td>
<td>• egid (Effective Group ID)</td>
</tr>
<tr>
<td><code>&lt;process&gt;</code></td>
<td>Valid attributes for the process field include:</td>
</tr>
<tr>
<td><code>&lt;process&gt;</code></td>
<td>• architecture</td>
</tr>
<tr>
<td><code>&lt;process&gt;</code></td>
<td>Valid architecture attribute values include:</td>
</tr>
<tr>
<td><code>&lt;process&gt;</code></td>
<td>0 For UNIX operating systems.</td>
</tr>
<tr>
<td><code>&lt;process&gt;</code></td>
<td>1 For Win32 operating systems.</td>
</tr>
<tr>
<td><code>&lt;process&gt;</code></td>
<td>Example:</td>
</tr>
<tr>
<td><code>&lt;process architecture=&quot;0&quot;&gt;</code></td>
<td>...</td>
</tr>
<tr>
<td><code>&lt;process&gt;</code></td>
<td><code>&lt;process&gt;</code></td>
</tr>
<tr>
<td><code>&lt;process&gt;</code></td>
<td><code>&lt;pid&gt;</code></td>
</tr>
<tr>
<td><code>&lt;process&gt;</code></td>
<td><code>&lt;eid&gt;</code></td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td>Types of processes. The process field can contain these fields:</td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td><code>pid</code> Process ID</td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td><code>eid</code> Effective user ID</td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td><code>uid</code> User Identifier (ID)</td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td><code>gid</code> Group Identifier (ID)</td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td><code>egid</code> Effective group ID</td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td>Example:</td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td><code>&lt;process architecture=&quot;unix&quot;&gt;</code></td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td>...</td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td><code>&lt;policy&gt;</code></td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td><code>&lt;name&gt;</code></td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td>Name of the policy attribute that you want to audit.</td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td>The name of the policy attribute matches the name that you specified in a list of attributes in the [aznapi-configuration] stanza of the appropriate configuration file. For example:</td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td>[aznapi-configuration] audit-attribute = real-traders-only</td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td>Example of name field for the policy field:</td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td><code>&lt;policy&gt;</code></td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td><code>&lt;name&gt;</code> real-traders-only&lt;/name&gt;</td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td><code>&lt;type&gt;</code>rule&lt;/type&gt;</td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td><code>&lt;/policy&gt;</code></td>
</tr>
</tbody>
</table>
Table 8. Names and descriptions for XML output fields  (continued)

<table>
<thead>
<tr>
<th>Output Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;type&gt;</code></td>
<td>Type of security policy being audited. Valid values include:</td>
</tr>
<tr>
<td></td>
<td>• ACL</td>
</tr>
<tr>
<td></td>
<td>• POP</td>
</tr>
<tr>
<td></td>
<td>• rule</td>
</tr>
<tr>
<td></td>
<td><code>pobj</code> refers to the type of security policy being used for the protected object.</td>
</tr>
<tr>
<td></td>
<td>Examples:</td>
</tr>
<tr>
<td><code>&lt;policy&gt;</code></td>
<td><code>&lt;name&lt;traders-pop&lt;/name&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;type&gt;POP&lt;/type&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/policy&gt;</code></td>
</tr>
</tbody>
</table>

| `<descr>`          | Description of the security policy. |
|                   | This field will not be filled in if no description was created for the policy. |
| `<policy>`         | `<name>traders-acl</name>` |
|                   | `<type>ACL</type>` |
|                   | `<descr>traders that have ACL security policies</descr>` |
|                   | `</policy>` |

| `<attribute>`      | Name of the access decision information (ADI) attribute to audit. An attribute can establish accountability by providing information to help identify potentially inappropriate access of assets. You can grant or deny access based on rules applied to attributes. |
|                   | The attribute field can contain these fields: |
|                   | • name |
|                   | • source |
|                   | • type |
|                   | • value |
|                   | Example: |
| `<attribute>`      | `<name>tagvalue_su-admin</name>` |
|                   | `<source>cred</source>` |
|                   | `<type>string</type>` |
|                   | `<value>test_customer_service_rep_1</value>` |
|                   | `</attribute>` |
Table 8. Names and descriptions for XML output fields (continued)

<table>
<thead>
<tr>
<th>Output Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;name&gt;...&lt;name&gt;</td>
<td>Name of the access decision information that you want to audit. The attribute can be for auditing either a user's credential if for the authn component or an app_context if for an azn component. The name of the authorization attribute matches the name that you specified in a list of attributes in the [aznapi-configuration] stanza of the appropriate configuration file. For example: [aznapi-configuration] audit-attribute = AZN_CRED_AUTH_METHOD Example of name field for the attribute field: &lt;attribute&gt; &lt;name&gt;AZN_CRED_AUTH_METHOD&lt;/name&gt; &lt;source&gt;credADI&lt;/source&gt; &lt;type&gt;string&lt;/type&gt; &lt;value&gt;su-forms&lt;/value&gt; &lt;/attribute&gt;</td>
</tr>
<tr>
<td>&lt;source&gt;...&lt;source&gt;</td>
<td>The source event can be one of the following: cred Applies to any Tivoli Access Manager component. app Applies only to an authorization (azn) component. credADI Applies only to the authorization (azn) component when evaluating a Boolean rule. appADI Applies only to the authorization (azn) component when evaluating a Boolean rule. engineADI Applies only to the authorization (azn) component when evaluating a Boolean rule. dynADI Applies only to the authorization (azn) component when evaluating a Boolean rule. If the ADI attribute is multi-valued, separate attribute element will be written for each value. Example: &lt;attribute&gt; &lt;name&gt;AZN_CRED_AUTH_METHOD&lt;/name&gt; &lt;source&gt;credADI&lt;/source&gt; &lt;type&gt;string&lt;/type&gt; &lt;value&gt;su-forms&lt;/value&gt; &lt;/attribute&gt;</td>
</tr>
</tbody>
</table>
### Table 8. Names and descriptions for XML output fields (continued)

<table>
<thead>
<tr>
<th>Output Field Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| <type>...<type>   | Type of security policy being audited. Valid values include:  
|                   | • string  
|                   | • ulong  
|                   | • obj  
|                   | If <type>obj</type>, the value will be name of the protected object.  
|                   | Example:  
|                   | <attribute>  
|                   | <name>AZN_CRED_AUTH_METHOD</name>  
|                   | <source>credADI</source>  
|                   | <type>string</type>  
|                   | <value>su-forms</value>  
|                   | </attribute> |
| <value>...<value>  | Value for the aznAPI attribute. If the ADI attribute is multi-valued, then a separate attribute element will be written for each value.  
|                   | Example:  
|                   | <attribute>  
|                   | <name>AZN_CRED_AUTH_METHOD</name>  
|                   | <source>credADI</source>  
|                   | <type>string</type>  
|                   | <value>su-forms</value>  
|                   | </attribute> |
Table 8. Names and descriptions for XML output fields (continued)

<table>
<thead>
<tr>
<th>Output Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;data&gt;</code></td>
<td>Event-specific data. The attribute fielded can contain this field:</td>
</tr>
<tr>
<td></td>
<td>• audit</td>
</tr>
<tr>
<td></td>
<td>Additional event-specific information is recorded in a free format data area at the end of the event record. For example, Table 10 on page 196 provides the data information that is returned when an authentication attempt fails.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Decoding the meaning of certain data values in the record might require an advanced knowledge of the Tivoli Access Manager code and architecture.</td>
</tr>
<tr>
<td></td>
<td>Command arguments are listed in the data field of the event record in their internal format. For example:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;data&gt;azn_id_get_creds&lt;/data&gt;</code></td>
</tr>
<tr>
<td></td>
<td>Note that commands that do not result in an effective change of state of the database (such as list and show) are never captured.</td>
</tr>
<tr>
<td></td>
<td>Example 1:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;event&gt;</code></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td><code>&lt;data&gt;</code></td>
</tr>
<tr>
<td></td>
<td>POST /pkmspasswd.form HTTP/1.1 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)</td>
</tr>
<tr>
<td></td>
<td><a href="https://c03comcrit2.somecompany.com/pkmspasswd">https://c03comcrit2.somecompany.com/pkmspasswd</a></td>
</tr>
<tr>
<td></td>
<td>&lt;/data&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/event&gt;</td>
</tr>
<tr>
<td></td>
<td>Example 2:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;data&gt;</code></td>
</tr>
<tr>
<td></td>
<td>&quot;2019&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;1002&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;pop1&quot;</td>
</tr>
<tr>
<td></td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td></td>
<td><code>&quot;=&quot;</code></td>
</tr>
<tr>
<td></td>
<td>&lt;/data&gt;</td>
</tr>
<tr>
<td><code>&lt;audit&gt;</code></td>
<td>Beginning and ending of an audit event. Valid attribute values include:</td>
</tr>
<tr>
<td></td>
<td>• Start</td>
</tr>
<tr>
<td></td>
<td>• Stop</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;event rev=&quot;1.2&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td><code>&lt;data&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;audit event=&quot;Start&quot;/&gt;</code></td>
</tr>
<tr>
<td></td>
<td>&lt;/data&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/event&gt;</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td><code>&lt;event rev=&quot;1.2&quot;&gt;</code></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td><code>&lt;data&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;audit event=&quot;Stop&quot;/&gt;</code></td>
</tr>
<tr>
<td></td>
<td>&lt;/data&gt;</td>
</tr>
</tbody>
</table>
|                   | </event>`
### Action codes for management commands

The action code identifies one of the Tivoli Access Manager `pdadmin` management commands. Table 9 relates the action code reference number for each management command. For example, the action code 13702 relates to the `POP_MODIFY` action command (the `pdadmin pop modify` command).

Command arguments are listed in the data section of the event record in their internal format. Note that commands that do not result in an effective change of state of the database (such as `list` and `show`) are never captured.

**Table 9. ACL management commands**

<table>
<thead>
<tr>
<th>ACL management commands</th>
<th>Reference number</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACL_LIST</td>
<td>13000</td>
</tr>
<tr>
<td>ACL_GET</td>
<td>13001</td>
</tr>
<tr>
<td>ACL_SET_LEGACY</td>
<td>13002</td>
</tr>
<tr>
<td>ACL_DELETE</td>
<td>13003</td>
</tr>
<tr>
<td>ACL_FIND</td>
<td>13005</td>
</tr>
<tr>
<td>ACTION_LIST</td>
<td>13006</td>
</tr>
<tr>
<td>ACTION_SET</td>
<td>13007</td>
</tr>
<tr>
<td>ACTION_DELETE</td>
<td>13008</td>
</tr>
<tr>
<td>ACTION_GROUPLIST</td>
<td>13009</td>
</tr>
<tr>
<td>ACTION_GROUPCREATE</td>
<td>13010</td>
</tr>
<tr>
<td>ACTION_GROUPDELETE</td>
<td>13011</td>
</tr>
<tr>
<td>ACTION_LISTGROUP</td>
<td>13012</td>
</tr>
<tr>
<td>ACTION_CREATEGROUP</td>
<td>13013</td>
</tr>
<tr>
<td>ACTION_DELETEGROUP</td>
<td>13014</td>
</tr>
<tr>
<td>ACL_CREATE</td>
<td>13020</td>
</tr>
<tr>
<td>ACL_SET</td>
<td>13021</td>
</tr>
<tr>
<td>ACL_CREATE_ATTR (deprecated, see 13134)</td>
<td>13150</td>
</tr>
</tbody>
</table>

**Object management commands**

<table>
<thead>
<tr>
<th>Object management commands</th>
<th>Reference number</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJ_GET</td>
<td>13100</td>
</tr>
<tr>
<td>OBJ_ACL_SET (deprecated)</td>
<td>13101</td>
</tr>
<tr>
<td>OBJ_GET_OBJ</td>
<td>13102</td>
</tr>
<tr>
<td>OBJSPC_CREATE</td>
<td>13103</td>
</tr>
<tr>
<td>OBJSPC_DELETE</td>
<td>13104</td>
</tr>
<tr>
<td>OBJSPC_LIST</td>
<td>13105</td>
</tr>
<tr>
<td>OBJ_CREATE</td>
<td>13106</td>
</tr>
<tr>
<td>OBJ_DELETE</td>
<td>13107</td>
</tr>
<tr>
<td>OBJ_MOD_SET_NAME</td>
<td>13110</td>
</tr>
<tr>
<td>OBJ_MOD_SET_DESC</td>
<td>13111</td>
</tr>
<tr>
<td>OBJ_MOD_SET_TYPE</td>
<td>13112</td>
</tr>
<tr>
<td>OBJ_MOD_SET_ISLF</td>
<td>13113</td>
</tr>
<tr>
<td>OBJ_MOD_SET_ISPOL</td>
<td>13114</td>
</tr>
<tr>
<td>OBJ_MOD_SET_ATTR</td>
<td>13115</td>
</tr>
</tbody>
</table>
Table 9. ACL management commands (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJ_MOD_DEL_ATTR</td>
<td>13116</td>
</tr>
<tr>
<td>OBJ_MOD_DEL_ATTRVAL</td>
<td>13117</td>
</tr>
<tr>
<td>OBJ_SHOW_ATTR</td>
<td>13118</td>
</tr>
<tr>
<td>OBJ_LIST_ATTR</td>
<td>13119</td>
</tr>
<tr>
<td>ACL_ATTACH</td>
<td>13120</td>
</tr>
<tr>
<td>ACL_DETACH</td>
<td>13121</td>
</tr>
<tr>
<td>ACL_MOD_SET_ATTR</td>
<td>13123</td>
</tr>
<tr>
<td>ACL_MOD_DEL_ATTR</td>
<td>13124</td>
</tr>
<tr>
<td>ACL_MOD_DEL_ATTRVAL</td>
<td>13125</td>
</tr>
<tr>
<td>ACL_SHOW_ATTR</td>
<td>13126</td>
</tr>
<tr>
<td>ACL_LIST_ATTR</td>
<td>13127</td>
</tr>
<tr>
<td>POP_MOD_SET_ATTR</td>
<td>13128</td>
</tr>
<tr>
<td>POP_MOD_DEL_ATTR</td>
<td>13129</td>
</tr>
<tr>
<td>POP_MOD_DEL_ATTRVAL</td>
<td>13130</td>
</tr>
<tr>
<td>POP_SHOW_ATTR</td>
<td>13131</td>
</tr>
<tr>
<td>POP_LIST_ATTR</td>
<td>13132</td>
</tr>
<tr>
<td>OBJ_SHOW_ATTRS</td>
<td>13133</td>
</tr>
<tr>
<td>ACL_SHOW_ATTRS</td>
<td>13134</td>
</tr>
<tr>
<td>POP_SHOW_ATTRS</td>
<td>13135</td>
</tr>
<tr>
<td>OBJ_SHOW_V417</td>
<td>13136</td>
</tr>
<tr>
<td>OBJ_LIST</td>
<td>13137</td>
</tr>
<tr>
<td>OBJ_LISTANDSHOW_V417</td>
<td>13138</td>
</tr>
<tr>
<td>OBJ_EXISTS (deprecated)</td>
<td>13139</td>
</tr>
<tr>
<td>OBJ_ACCESS_CHECK</td>
<td>13140</td>
</tr>
<tr>
<td>OBJ_SHOW</td>
<td>13141</td>
</tr>
<tr>
<td>OBJ_LISTANDSHOW</td>
<td>13142</td>
</tr>
<tr>
<td>ACL_CREATE_ATTR (deprecated, see 13134)</td>
<td>13150</td>
</tr>
</tbody>
</table>

Server management commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVER_GET</td>
<td>13200</td>
</tr>
<tr>
<td>SERVER_RESTORE</td>
<td>13201</td>
</tr>
<tr>
<td>SERVER_DELETE (deprecated)</td>
<td>13202</td>
</tr>
<tr>
<td>SERVER_LIST</td>
<td>13203</td>
</tr>
<tr>
<td>SERVER_PERFORMTASK</td>
<td>13204</td>
</tr>
<tr>
<td>SERVER_GETTASKLIST</td>
<td>13205</td>
</tr>
<tr>
<td>SERVER_REPLICATE</td>
<td>13206</td>
</tr>
<tr>
<td>SERVER_ACTION</td>
<td>13207</td>
</tr>
<tr>
<td>SERVER_STATUS_GET</td>
<td>13208</td>
</tr>
<tr>
<td>SERVER_ENABLE (deprecated)</td>
<td>13209</td>
</tr>
<tr>
<td>SERVER_DISABLE (deprecated)</td>
<td>13210</td>
</tr>
<tr>
<td>Administration, user, and group management commands</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>ADMIN_SHOWCONF</td>
<td>13400</td>
</tr>
<tr>
<td>USER_CREATE</td>
<td>13401</td>
</tr>
<tr>
<td>USER_IMPORT</td>
<td>13402</td>
</tr>
<tr>
<td>USER_MODDESC</td>
<td>13403</td>
</tr>
<tr>
<td>USER_MDPWD</td>
<td>13404</td>
</tr>
<tr>
<td>USER_MDAUTHMECH</td>
<td>13405</td>
</tr>
<tr>
<td>USER_MODACCVALID</td>
<td>13406</td>
</tr>
<tr>
<td>USER_MODPWDVALID</td>
<td>13407</td>
</tr>
<tr>
<td>USER_DELETE</td>
<td>13408</td>
</tr>
<tr>
<td>USER_SHOWGROUPS</td>
<td>13409</td>
</tr>
<tr>
<td>USER_SHOW</td>
<td>13410</td>
</tr>
<tr>
<td>USER_SHOWDN</td>
<td>13411</td>
</tr>
<tr>
<td>USER_LIST</td>
<td>13412</td>
</tr>
<tr>
<td>USER_LISTDN</td>
<td>13413</td>
</tr>
<tr>
<td>GROUP_CREATE</td>
<td>13414</td>
</tr>
<tr>
<td>GROUP_IMPORT</td>
<td>13415</td>
</tr>
<tr>
<td>GROUP_MODDESC</td>
<td>13416</td>
</tr>
<tr>
<td>GROUP_MODADD</td>
<td>13417</td>
</tr>
<tr>
<td>GROUP_MODREMOVE</td>
<td>13418</td>
</tr>
<tr>
<td>GROUP_DELETE</td>
<td>13419</td>
</tr>
<tr>
<td>GROUP_SHOW</td>
<td>13420</td>
</tr>
<tr>
<td>GROUP_SHOWDN</td>
<td>13421</td>
</tr>
<tr>
<td>GROUP_LIST</td>
<td>13422</td>
</tr>
<tr>
<td>GROUP_LISTDN</td>
<td>13423</td>
</tr>
<tr>
<td>GROUP_SHOWMEMB</td>
<td>13424</td>
</tr>
<tr>
<td>USER_MODGSOUSER</td>
<td>13425</td>
</tr>
<tr>
<td>USER_SET (deprecated)</td>
<td>13426</td>
</tr>
<tr>
<td>GROUP_SET (deprecated)</td>
<td>13427</td>
</tr>
<tr>
<td>GROUP_MODADD2</td>
<td>13428</td>
</tr>
<tr>
<td><strong>GSO resource commands</strong></td>
<td></td>
</tr>
<tr>
<td>GSO_Resource_CREATE</td>
<td>13500</td>
</tr>
<tr>
<td>GSO_Resource_DELETE</td>
<td>13501</td>
</tr>
<tr>
<td>GSO_Resource_LIST</td>
<td>13502</td>
</tr>
<tr>
<td>GSO_Resource_SHOW</td>
<td>13503</td>
</tr>
<tr>
<td><strong>GSO resource credential commands</strong></td>
<td></td>
</tr>
<tr>
<td>GSO_Resource_CRED_CREATE</td>
<td>13504</td>
</tr>
<tr>
<td>GSO_Resource_CRED_DELETE</td>
<td>13505</td>
</tr>
<tr>
<td>GSO_Resource_CRED_MODIFY</td>
<td>13506</td>
</tr>
<tr>
<td>GSO_Resource_CRED_LIST</td>
<td>13507</td>
</tr>
<tr>
<td>GSO_Resource_CRED_SHOW</td>
<td>13508</td>
</tr>
</tbody>
</table>

Chapter 18. XML output for logging and auditing logs 193
<table>
<thead>
<tr>
<th>GSO resource group commands</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GSO_RESOURCE_GROUP_CREATE</td>
<td>13509</td>
</tr>
<tr>
<td>GSO_RESOURCE_GROUP_DELETE</td>
<td>13510</td>
</tr>
<tr>
<td>GSO_RESOURCE_GROUP_ADD</td>
<td>13511</td>
</tr>
<tr>
<td>GSO_RESOURCE_GROUP_REMOVE</td>
<td>13512</td>
</tr>
<tr>
<td>GSO_RESOURCE_GROUP_LIST</td>
<td>13513</td>
</tr>
<tr>
<td>GSO_RESOURCE_GROUP_SHOW</td>
<td>13514</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Policy commands</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>POLICY_SET_MAX_LOGIN_FAILURES</td>
<td>13600</td>
</tr>
<tr>
<td>POLICY_GET_MAX_LOGIN_FAILURES</td>
<td>13601</td>
</tr>
<tr>
<td>POLICY_SET_DISABLE_TIME_INTERVAL</td>
<td>13602</td>
</tr>
<tr>
<td>POLICY_GET_DISABLE_TIME_INTERVAL</td>
<td>13603</td>
</tr>
<tr>
<td>POLICY_SET_MAX_ACCOUNT_AGE</td>
<td>13604</td>
</tr>
<tr>
<td>POLICY_GET_MAX_ACCOUNT_AGE</td>
<td>13605</td>
</tr>
<tr>
<td>POLICY_SET_ACCOUNT_EXPIRY_DATE</td>
<td>13606</td>
</tr>
<tr>
<td>POLICY_GET_ACCOUNT_EXPIRY_DATE</td>
<td>13607</td>
</tr>
<tr>
<td>POLICY_SET_MAX_INACTIVITY_TIME</td>
<td>13608</td>
</tr>
<tr>
<td>POLICY_GET_MAX_INACTIVITY_TIME</td>
<td>13609</td>
</tr>
<tr>
<td>POLICY_GET_ACCOUNT_CREATION_DATE</td>
<td>13610</td>
</tr>
<tr>
<td>POLICY_GET_LAST_LOGIN_ATTEMPT_DATE</td>
<td>13611</td>
</tr>
<tr>
<td>POLICY_SET_MAX_PASSWORD_AGE</td>
<td>13612</td>
</tr>
<tr>
<td>POLICY_GET_MAX_PASSWORD_AGE</td>
<td>13613</td>
</tr>
<tr>
<td>POLICY_SET_MIN_PASSWORD_AGE</td>
<td>13614</td>
</tr>
<tr>
<td>POLICY_GET_MIN_PASSWORD_AGE</td>
<td>13615</td>
</tr>
<tr>
<td>POLICY_SET_MAX_PASSWORD_REPEATED_CHARS</td>
<td>13616</td>
</tr>
<tr>
<td>POLICY_GET_MAX_PASSWORD_REPEATED_CHARS</td>
<td>13617</td>
</tr>
<tr>
<td>POLICY_SET_MIN_PASSWORD_ALPHAS</td>
<td>13618</td>
</tr>
<tr>
<td>POLICY_GET_MIN_PASSWORD_ALPHAS</td>
<td>13619</td>
</tr>
<tr>
<td>POLICY_SET_MIN_PASSWORD_NON_ALPHAS</td>
<td>13620</td>
</tr>
<tr>
<td>POLICY_GET_MIN_PASSWORD_NON_ALPHAS</td>
<td>13621</td>
</tr>
<tr>
<td>POLICY_SET_MIN_PASSWORD_DIFFERENT_CHARS</td>
<td>13622</td>
</tr>
<tr>
<td>POLICY_GET_MIN_PASSWORD_DIFFERENT_CHARS</td>
<td>13623</td>
</tr>
<tr>
<td>POLICY_SET_PASSWORD_SPACES</td>
<td>13624</td>
</tr>
<tr>
<td>POLICY_GET_PASSWORD_SPACES</td>
<td>13625</td>
</tr>
<tr>
<td>POLICY_SET_MIN_PASSWORD_LENGTH</td>
<td>13626</td>
</tr>
<tr>
<td>POLICY_GET_MIN_PASSWORD_LENGTH</td>
<td>13627</td>
</tr>
<tr>
<td>POLICY_SET_MIN_PASSWORD_REUSE_TIME</td>
<td>13628</td>
</tr>
<tr>
<td>POLICY_GET_MIN_PASSWORD_REUSE_TIME</td>
<td>13629</td>
</tr>
<tr>
<td>POLICY_GET_PASSWORD_FAILURES</td>
<td>13630</td>
</tr>
<tr>
<td>POLICY_GET_LAST_PASSWORD_CHANGE_DATE</td>
<td>13631</td>
</tr>
<tr>
<td>POLICY_SET_NUMBER_WARN_DAYS</td>
<td>13632</td>
</tr>
<tr>
<td>Table 9. ACL management commands (continued)</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>POLICY_GET_NUMBER_WARN_DAYS</td>
<td>13633</td>
</tr>
<tr>
<td>POLICY_SET_PASSWORD_REUSE_NUM</td>
<td>13634</td>
</tr>
<tr>
<td>POLICY_GET_PASSWORD_REUSE_NUM</td>
<td>13635</td>
</tr>
<tr>
<td>POLICY_SET_TOD_ACCESS</td>
<td>13636</td>
</tr>
<tr>
<td>POLICY_GET_TOD_ACCESS</td>
<td>13637</td>
</tr>
<tr>
<td>POLICY_GET_ALL_POLICY</td>
<td>13638</td>
</tr>
<tr>
<td><strong>POP commands</strong></td>
<td></td>
</tr>
<tr>
<td>POP_CREATE</td>
<td>13700</td>
</tr>
<tr>
<td>POP_DELETE</td>
<td>13701</td>
</tr>
<tr>
<td>POP_MODIFY</td>
<td>13702</td>
</tr>
<tr>
<td>POP_SHOW</td>
<td>13703</td>
</tr>
<tr>
<td>POP_LIST</td>
<td>13704</td>
</tr>
<tr>
<td>POP_ATTACH</td>
<td>13705</td>
</tr>
<tr>
<td>POP_DETACH</td>
<td>13706</td>
</tr>
<tr>
<td>POP_FIND</td>
<td>13707</td>
</tr>
<tr>
<td><strong>Configuration commands</strong></td>
<td></td>
</tr>
<tr>
<td>CFG_CONFIG</td>
<td>13800</td>
</tr>
<tr>
<td>CFG_UNCONFIG</td>
<td>13801</td>
</tr>
<tr>
<td>CFG_RENEWCERT</td>
<td>13802</td>
</tr>
<tr>
<td>CFG_SETPORT</td>
<td>13803</td>
</tr>
<tr>
<td>CFG_SETLISTENING</td>
<td>13804</td>
</tr>
<tr>
<td>CFG_SETKEYRINGPWD</td>
<td>13805</td>
</tr>
<tr>
<td>CFG_SETSSLTIMEOUT</td>
<td>13806</td>
</tr>
<tr>
<td>CFG_SETAPPLCERT</td>
<td>13807</td>
</tr>
<tr>
<td>CFG_ADDREPLICA</td>
<td>13808</td>
</tr>
<tr>
<td>CFG_CHGREPLICA</td>
<td>13809</td>
</tr>
<tr>
<td>CFG_RMREPlica</td>
<td>13810</td>
</tr>
<tr>
<td>CFG_GETVALUE</td>
<td>13811</td>
</tr>
<tr>
<td>CFG_SETVALUE</td>
<td>13812</td>
</tr>
<tr>
<td>CFG_RMVALUE</td>
<td>13813</td>
</tr>
<tr>
<td>CFG_SETSVRPWD</td>
<td>13814</td>
</tr>
<tr>
<td><strong>Domain commands</strong></td>
<td></td>
</tr>
<tr>
<td>DOMAIN_CREATE</td>
<td>13900</td>
</tr>
<tr>
<td>DOMAIN_DELETE</td>
<td>13901</td>
</tr>
<tr>
<td>DOMAIN_MODIFY_DESC</td>
<td>13902</td>
</tr>
<tr>
<td>DOMAIN_SHOW</td>
<td>13903</td>
</tr>
<tr>
<td>DOMAIN_LIST</td>
<td>13904</td>
</tr>
</tbody>
</table>

Chapter 18. XML output for logging and auditing logs  195
### Table 9. ACL management commands (continued)

<table>
<thead>
<tr>
<th>Rule commands</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTHZRULE_CREATE</td>
<td>13950</td>
</tr>
<tr>
<td>AUTHZRULE_DELETE</td>
<td>13951</td>
</tr>
<tr>
<td>AUTHZRULE_MODIFYTEXT</td>
<td>13952</td>
</tr>
<tr>
<td>AUTHZRULE_MODIFYREASON</td>
<td>13953</td>
</tr>
<tr>
<td>AUTHZRULE_MODIFYDESC</td>
<td>13954</td>
</tr>
<tr>
<td>AUTHZRULE_SHOW</td>
<td>13955</td>
</tr>
<tr>
<td>AUTHZRULE_LIST</td>
<td>13956</td>
</tr>
<tr>
<td>AUTHZRULE_ATTACH</td>
<td>13957</td>
</tr>
<tr>
<td>AUTHZRULE_DETACH</td>
<td>13958</td>
</tr>
<tr>
<td>AUTHZRULE_FIND</td>
<td>13959</td>
</tr>
<tr>
<td>AUTHZRULE_MOD_SET_ATTR</td>
<td>13960</td>
</tr>
<tr>
<td>AUTHZRULE_MOD_DEL_ATTR</td>
<td>13961</td>
</tr>
<tr>
<td>AUTHZRULE_MOD_DEL_ATTRVAL</td>
<td>13962</td>
</tr>
<tr>
<td>AUTHZRULE_SHOW_ATTRS</td>
<td>13963</td>
</tr>
<tr>
<td>AUTHZRULE_SHOW_ATTR</td>
<td>13964</td>
</tr>
<tr>
<td>AUTHZRULE_LIST_ATTR</td>
<td>13965</td>
</tr>
</tbody>
</table>

### Data output for authentication errors

Table 10 lists the authentication error codes and the `<data>` output field element structures that are returned when an authentication attempt fails:

#### Table 10. Authentication errors

<table>
<thead>
<tr>
<th>Error type</th>
<th>Error code (in hex)</th>
<th>Error code (in decimal)</th>
<th>XML generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password failure</td>
<td>132120c8</td>
<td>320938184</td>
<td>&lt;data&gt;&lt;passwordFailure: user&lt;/data&gt;</td>
</tr>
<tr>
<td>Account lock-out</td>
<td>13212132</td>
<td>320938290</td>
<td>&lt;data&gt;&lt;accountLockOut: user&lt;/data&gt;</td>
</tr>
<tr>
<td>General failure</td>
<td>All others</td>
<td>All others</td>
<td>&lt;data&gt;&lt;username&gt;user&lt;/username&gt;&lt;/data&gt;</td>
</tr>
</tbody>
</table>
Chapter 19. Capturing logging and auditing events

Tivoli Access Manager provides two methods of capturing audit events. One method is used to capture audit events for the current version of Tivoli Access Manager and the other method is used to capture audit events for legacy purposes. Use this method to capture audit events for the current version of Tivoli Access Manager.

For auditing and other serviceability purposes, apart from some messages that are produced when starting a program, all messages generated by Tivoli Access Manager are created in a structured hierarchy of Tivoli Access Manager events. The orderly categorization of events within this hierarchy allows runtime associations to be made between classes of events and the log agents that are to be used to record those events.

This chapter describes the configuration stanza entries for each type of log agent. This chapter contains the following sections:

- “Log agents”
- “Categories of events”
- “Configuration parameters for EventPool” on page 198
- “Event logging tasks” on page 211

Log agents

With event logging, the concept of a log agent includes capturing events that are redirected to destinations other than the local file system. Event logging uses these types of log agents, each representing a destination for the event record:

- Console log agent
- File log agent
- Pipe log agent
- Remote log agent

Categories of events

The event hierarchy is built up dynamically during program execution. While some well-known event categories can be expected to be present when running a Tivoli Access Manager program, other categories can be program specific, and some can be transient.

Program–specific event category

A specific event category is identified by a dot-separated list of names. The first level of names within the category has special significance. This top-level category name also might correspond to events previously associated with legacy Tivoli Access Manager log files described in “Overview of auditing” on page 219.

For example, assume that the event category name is constructed as follows:

domain_category.sub_category.sub_category,...sub_category
Implementation note: For efficiency, an event is not generally created if there are no log agents subscribed to record events of that category. In the case that an event is generated and there are no log agents subscribed to record it, the event is discarded.

Event pool hierarchy
This event logging section describes how you can associate log agents with a point in the event pool hierarchy to record events. For example:

```
[aznapi-configuration]
logcfg = EventPool:file queue_size=number,
       hi_water=number, flush_interval=number_seconds
```

A description of the characteristics of all possible events within the hierarchy is not provided. For descriptions of well-known events such as those generated for auditing, refer to the appropriate product-specific documentation.

Generally, the event pool hierarchy is similar to the following:

![Event Pool Diagram]

Figure 29. Event pool hierarchy

Configuration parameters for EventPool
In addition to the backward-compatible, legacy method of auditing and sending events to a audit trail log file (auditcfg = stanza entry), you can configure the capture of Tivoli Access Manager events by using the logcfg = stanza entry.

To configure the capture of Tivoli Access Manager events, you can specify appropriate stanza entries in the [aznapi-configuration] stanza of a server’s configuration file (server_name.conf).
Use the logcfg parameter to specify event logging. The parameters and values vary, depending on the category, the destination of events, and the type of event logging that you want to perform.

The general format of the entry (entered on one line) that is needed for event logging to the log agents is:

```
[aznapi-configuration]
logcfg = category:\{stdout|stderr|file|pipe|remote\}
[\{param=[value]\} [param=[value]]]
```

To enable the recording of Tivoli Access Manager events using the interface, you must associate a logging destination with a category of events in the event pool. Currently four types of destinations are supported for the capture of events:
- Console (stdout|stderr) log agent
- File log agent
- Pipe log agent
- Remote log agent.

Refer to any of the following event logging destinations for more specific examples for the logcfg parameter:

Options for these log agent types can be specified in any order and are generally optional. Valid options for each log agent type are described below. In a configuration entry, the option names are case-insensitive and can be abbreviated to any shortened length of the full option name that remains unique.

The category name can point to any node in the event pool hierarchy. For example, consider the following simplified form:

```
[aznapi-configuration]
logcfg = category:log-agent
```

Capture of events for a category is inclusive of all subcomponents in the hierarchy. That is, a `foo.bar.fred` event also is captured at the `foo.bar` category.

You can attach multiple log agents to the same category. For example, the following configuration copies authorization audit events to a file and relays them to a program listening on a pipe:

```
[aznapi-configuration]
logcfg = audit.azn:file path=/var/PolicyDirector/log/audit.azn
logcfg = audit.azn:pipe path=/bin/analyse.exe
```

The following diagram depicts the relationships between steps in the logging process. The top third of the diagram represents the code of a Tivoli Access Manager server. The programmer added probe points to the code where events of specific types might be generated. Generated events are then submitted to the server’s event pool for possible recording through a point of capture (the sink), which defines the events category.

At runtime, a user can subscribe a log agent at any point in the event pool hierarchy to selectively record events generated at the program’s probe points. This is depicted in the middle band of the diagram.

One log agent that you can subscribe to for capturing events is a remote log client. This client forwards the selected events to a remote `pdacld` server. The bottom band of the diagram depicts this remote server. Note that the bottom band is
essentially the same as the top band with the relayed events placed in the event pool at the **pdacld** remote probe points.

*Figure 30. Application-Specific Probe Points*
**EventPool**

Events are passed to subscribed log agents asynchronously to the application-level requests that construct the events for recording. Events initially pass through a common propagation queue before they are fanned out to the variously subscribed log agents.

The servicing profile of this propagation queue is configurable. To configure the propagation queue, you must specify an abridged format logcfg stanza entry. The shortened configuration entry uses EventPool as the category name and specifies queuing options without giving a log destination type.

You should manage the propagation queue to support the configuration of the log agents. For example, when limiting the amount of memory used to queue log requests for a remote log agent, you should constrain the event propagation queue_size also:

```bash
[aznapi-configuration]
logcfg = EventPool:file queue_size=number,hi_water=number,
       flush_interval=number_seconds

logcfg = EventPool:remote buffer_size=number,path=pathname,
       server=hostname,queue_size=number
```

**queue_size**

To control the amount of memory that can be consumed by events on the propagation queue, you can set a limit for the maximum size the queue is allowed to grow to. If the maximum size is reached when a new event is generated, the thread attempting to queue it is blocked until space is available in the queue. This has the effect of throttling back performance of the event producing thread to the speed of the logging threads, if it cannot keep up.

This example shows how to specify the maximum number of events to queue in memory:

```bash
[aznapi-configuration]
logcfg = EventPool:queue_size=number,
       hi_water=number,flush_interval=num_seconds
```

The default value for queue size is 0. A zero queue size indicates that no limit is enforced on the growth of the event queue. Keep in mind that using the default value can allow the unprocessed event queue to grow to an unmanageable size when events are produced at a rate faster than the subscribed log agents can clear them.

**hi-water**

Processing of the event queue is scheduled regularly at the configured flush interval. It is also triggered asynchronously by the queue size reaching a high water mark.

This example shows how to specify the event queue high water mark:

```bash
[aznapi-configuration]
logcfg = EventPool:queue_size=number,
       hi_water=number,flush_interval=num_seconds
```
The default value for the high water mark is 1024. If you specify a value for queue_size, but not a value for hi_water, the default value for the high water mark is calculated as two-thirds of the maximum configured queue size.

For example, if the maximum queue size is 0, the high water mark will be set to a default of 100. If the event queue high water mark is set to 1, every event queued will be relayed to any subscribed log agents as soon as possible.

Note that setting a low value for the high water mark can have an adverse effect on overall performance. See the IBM Tivoli Access Manager for e-business Performance Tuning Guide for more information.

flush_interval

Use the flush interval to specify a limit on the time an event waits in the propagation queue before it is forwarded to the log agents. If events are being generated at a slow rate that does not trigger handling by reaching the high water mark in a timely manner, events are flushed from the propagation queue at this frequency.

This example shows how to set the frequency for flushing log file buffers:

```
[aznapi-configuration]
logcfg = EventPool:queue_size=number,hi_water=number,
flush_interval=num_seconds
```

The default value for the flush interval is 10 seconds. A flush interval of 0 is not allowed. Specifying a value of 0 results in the queue being flushed every 600 seconds.

---

**Configuration parameters for console log agent**

Logging to the console is the easiest event logging option to configure. Simply associate an output destination of standard out or standard error with the category of events in the event pool to capture:

```
[aznapi-configuration]
logcfg = category: {stdout|stderr}
```

Logging to the console does not itself use any queuing. The events are written to the console as they are received from the propagation queue. However, note that events might be delayed in the propagation queue, depending on its queue settings.

If you are using console output and running a server in the foreground for debugging purposes, you might want to set the propagation queue settings accordingly. For example, set the hi_water option to a low value.

Example stdout and stderr configurations follow.

**stdout**

To capture all audit output to standard out, specify the following stanza entry in the configuration file:

```
[aznapi-configuration]
logcfg = category: stdout
```
stderr

To capture only authorization (azn) audit events to standard error, specify the following stanza entry in the configuration file:

```plaintext
[aznapi-configuration]
logcfg = category:stderr
```

**Configuration parameters for a file log agent**

To record events in a file, specify a log file configuration as follows:

```plaintext
[aznapi-configuration]
logcfg = category:file path=file_pathname, flush_interval=num_seconds,
rollover_size=number, log_id=logid, queue_size=number, hi_water=number,
buffer_size=number, mode={text|binary}
```

Argument names can be abbreviated to any unambiguous prefix of the full name. For example, the argument hi_water= can be abbreviated to hi=.

A file is only opened once. If multiple configuration entries exist to selectively capture events at different points of the event pool hierarchy to the same file, the file opens according to the options found in the first configuration entry processed.

After a file has been opened, further file configurations can simply use the following shorthand notation to record events to the same file:

```plaintext
[aznapi-configuration]
logcfg = category:file log_id=logid
```

Because writing to file can be a slow operation relative to the tasks generating events, events are posted to a file log agent through a second level of queuing. This second level of event queuing is configured in a similar manner to the central event propagation queue, but has different default values.

**path**

The path specifies the name and location of a log file. There is no default value for the file path name because the log_id value takes precedence. An example path value for the WebSEAL audit trail file on UNIX is as follows:

```plaintext
[aznapi-configuration]
logcfg = category:file
path=/var/pdweb/log/audit.log
```

The directory portion of this path name must exist. The log file is created if it does not already exist.

**logcfg**

An open log file is associated with a short name identifier to facilitate the recording of events from different categories to the same file.

Use the log_id option to set the log file identifier (ID) explicitly; otherwise, it is given a default value. If the path= option is specified, the default value is the configured path name. If path= is not specified, the log ID defaults to the domain component of the event category being captured. For example:

```plaintext
logcfg = audit.azn:file
```

implies

```plaintext
log_id=audit
```
To capture events to a common file, set the log file ID to a suitable value in a fully optioned file configuration. Subsequently, use the shorthand configuration variant to capture events from additional categories as shown:

```
[aznapi-configuration]
logcfg = audit.azn:file path=/opt/PolicyDirector/log/audit.log,
rollover_size=1,flush_interval=20,log_id=audit,
...
logcfg = audit.authn:file log_id=audit
```

Because of the default rules, this configuration is also equivalent to the following specification:

```
[aznapi-configuration]
logcfg = audit.azn:file path=/opt/PolicyDirector/log/audit.log,
rollover_size=1,
...
logcfg = audit.authn:file
```

If you construct a configuration where the log ID value does not match any open log file, no events are captured. For example, the following configuration does not record any events because the configuration line that initializes the log file has been commented out:

```
[aznapi-configuration]
#logcfg = audit.azn:file path=/tmp/azn.log,log_id=azn,
...
logcfg = audit.authn:file log_id=auth
```

**rollover_size**

Use the rollover_size option to specify the maximum size to which a log file can grow. This option has the following default value (in bytes):

```
[aznapi-configuration]
logcfg = audit.azn:file
...
rollover_size=2000000
```

When the size of a log file reaches the specified value, known as its rollover threshold, the existing file is backed up to a file of the same name with an appended current date and time stamp. A new log file is then started.

The various possible rollover size values are interpreted as follows:

- If the rollover size value is less than zero (< 0), a new log file is created with each invocation of the process and every 24 hours from that instance.
- If the rollover size value is equal to zero (= 0), no rollovers are performed, and the log file grows indefinitely. If a log file already exists, new data is appended to it.
- If the rollover size value is greater than zero (> 0), a rollover is performed when a log file reaches the configured threshold value. If a log file already exists at startup, new data is appended to it.

**buffer_size**

To reduce memory fragmentation and improve the performance of writing to a file, rather than queuing many small events individually to the file log agent, events can be buffered into blocks of a nominated size before queuing for writing. The
buffer_size option specifies the maximum size message that the program attempts to construct by combining smaller events into a large buffer.

Buffers consist of only an integer number of events; events are not split across buffers. If any individual event exceeds that maximum configured size, the large event is recorded in a buffer of its own, exceeding the configured value.

```
[aznapi-configuration]
logcfg = audit.azn:file
...
buffer_size=0
number
...  
```

The default buffer size for logging to a file is 0 bytes. This value prevents buffering and each event is handled individually.

If a value is specified for the buffer_size, events are packed into buffers of that size before queuing to the file log agent.

For example, if the buffer_size value is set to 2 KB and events are assumed to be about 256 bytes, around ten events are packed into each buffer written to the file. This reduces the number of disk input/outputs (I/Os) that are made while logging to 10 percent of the equivalent non-buffering case.

Note that a default queue size of 200 with a buffer_size of 2 KB also consumes around ten times the memory of a default configuration that did no buffering (assuming an event size of around 200 bytes). This is because the maximum queue size value has not been changed, but the size of events being queued has increased tenfold.

**queue_size**

There is a delay between events being placed on the queue and the file log agent removing them. The queue_size option specifies the maximum size to which the queue is allowed to grow. If the maximum size is reached when a new event is ready to be placed on the queue, the requesting thread is blocked until space is available in the queue. This has the effect of throttling back performance of the event propagation thread to the speed of the file logging thread if it cannot keep up.

Limiting the queue size for the log agent should be configured in conjunction with setting the queue size for the central event propagation queue. Unless the event propagation queue_size is constrained appropriately, memory usage can still grow without bounds.

```
[aznapi-configuration]
logcfg = audit.azn:file
...
queue_size=number_events,
...  
```

The default value for queue_size is 0. A zero queue size means that no limit is enforced on the growth of the unprocessed event queue. Correspondingly, the event propagation thread is not constrained by the speed of the logging thread. Keep in mind that using the default can result in the unrecorded event queue growing to an unmanageable size, if events are being generated faster than they can be recorded to file.
**hi_water**

Processing of the event queue is scheduled regularly at the configured flush interval. It also is triggered asynchronously by the queue size reaching a high water mark on the event queue.

```ini
[aznapi-configuration]
logcfg = audit.azn:file
...
hi_water=number,
...
```

The default value for hi_water is two-thirds of the maximum configured queue size. If the maximum queue size is zero, the high water mark is set to a default of 100.

The transaction rates and the values of these options determine the maximum amount of memory that is consumed by enabling event logging to file.

If the event queue high water mark is set to 1, every event queued is relayed to the log agent as soon as possible. This setting is not optimal, although you might want to use it if you want to ensure events get to disk as fast as possible, at the expense of overall performance.

**flush_interval**

The flush_interval option is a multiuse option. The logging to file flush_interval option has the following default value in seconds.

```ini
[aznapi-configuration]
logcfg = audit.azn:file
...
flush_interval={0|<0|number_seconds},
...
```

Log files are written to buffered data streams. To ensure stream buffers are flushed to disk regularly, the frequency with which the server asynchronously forces a flush of the file stream to disk is configurable using the flush_interval option.

A flush interval of 0 is not allowed. If you specify a value of 0, it will result in the value 600 seconds being used.

If you specify a negative value (<0), the absolute value will be used as the asynchronous flush frequency, but a stream flush is also forced synchronously after every record is written.

If events are being consolidated into large buffers by specifying a value for the buffer_size option, the flush_interval option also might affect the size of buffer written. If there is a partially filled buffer in memory when a flush is scheduled, that buffer is also queued for writing before it completes the buffer fill.

The event queue is triggered for processing at the flush interval rate. This prevents events waiting to be processed for longer than the scheduled flush time when the queue high water mark is not reached between scheduled flushes.
mode

Use the mode option to open a file in either text or binary mode. For example:

```plaintext
[aznapi-configuration]
logcfg = audit.azn:file
...
mode={text|binary},
...
```

Text mode is deprecated on UNIX platforms and has no effect. On Microsoft Windows 32–bit platforms, opening a file in text mode enables end-of-line character translations in the log file. Binary mode on a Windows platform writes the log file in a UNIX compatible format.

---

Configuration parameters for pipe log agent

Use the pipe option to write output to the standard input of another program. For example:

```plaintext
[aznapi-configuration]
logcfg = category:pipe path=program_pathname, 
queue_size=number, 
hi_water=number, 
flush_interval=number_seconds
```

Argument names can be abbreviated to any unambiguous prefix of the full name. For example, the argument hi_water= can be abbreviated to hi=.

The named program must exist and be executable. The administrator is responsible for ensuring the security of the program that is to be run.

Each occurrence of a pipe agent in the configuration file invokes a new copy of the pipe program. Unlike logging to file, piped events are not multiplexed from different category capture points to a single copy of the program.

path

Use the path option to specify the location of the program, which receives the log output on standard input. For example:

```plaintext
[aznapi-configuration]
logcfg = category:pipe
path=/opt/risk_analyser/bin/my_log_watcher
```

Note that there is no default value for the path name.

queue_size

Configure the pipe logging event queue management in the same way that you configure logging to file. The queue_size option has similar meaning to the option described for file logging.

```plaintext
queue_size={0|number_events}
```
**hi_water**

Configure the pipe logging event queue management in the same way that you configure logging to file. The `hi_water` option has similar meaning to the option described for file logging.

`hi_water={0|1|number}`

**flush_interval**

Configure the pipe logging event queue management in the same way that you configure logging to file. The `flush_interval` option has similar meaning to the option described for file logging.

`flush_interval={0|<0|number_seconds}`

---

### Configuration parameters for remote log agent

Use the remote option to send events to a remote server for recording. For example:

```
[aznapi-configuration]
logcfg = category:remote buffer_size=size,
        compress={yes|no},error_retry=timeout,path=name,
        flush_interval=number_seconds,rebind_retry=timeout,
        server=hostname, port=number, dn=identity,
        queue_size=number, hi_water=number
```

Argument names can be abbreviated to any unambiguous prefix of the full name. For example, the argument `hi_water=` can be abbreviated to `hi=`.

Requests to log an event remotely are accepted on a best effort basis only. If the remote server is not available, captured events are cached locally and relayed at a later date, if and when the remote server becomes available.

Only one remote logging connection is established to a remote server. If multiple configuration entries are made to selectively capture events at different points of the event pool hierarchy to the same remote server, the remote connection is established according to the options of the first remote configuration entry processed. Multiple remote connections can be configured to log to different remote servers.

Events received at the remote server are placed in the event pool of that server in a different location from where they were originally captured on the client system. All events entering a host through the remote logging service are placed in a category constructed in the following manner:

```
remote.client-category-domain.hostname.program
```

The internal logging code on the client provides the host name by using the `gethostname()` system call. The receiving host can always resolve the host name of the sender because `gethostname()` is always available on the client. Therefore, there are no special requirements for network host name resolution. This call returns the internal host name of the local host—always in short name form.

For example, all audit events logged remotely from program `pdmgrd` on host `amazon` appear on the remote log server under pool `remote.audit.amazon.pdmgrd`. Appearing under one pool allows for the remote server to selectively record events in a variety of destinations using standard configurations. All audit events from host `amazon` can be recorded centrally on host `timelord` by configurations such as the following examples.
On host amazon to relay events remotely, you might use this example:

```bash
[aznapi-configuration]
logcfg = audit:remote buffer=2000,compress=y,error=2, \npath=/opt/PolicyDirector/log/remote.cache, rebind=600, server=timelord, port=7136
```

On host timelord to record events to file, you might use:

```bash
[aznapi-configuration]
logcfg = remote.audit:file path=consolidated_audit.log
logcfg = remote.audit.amazon.pdmgrd:file path=amazon_pdmgrd_audit.log
```

**buffer_size**

To reduce network traffic, events are buffered into blocks of the nominated size before relaying to the remote server. The `buffer_size` option specifies the maximum size message that the local program attempts to construct by combining smaller events into a large buffer. Buffers consist only of an integral number of events; events are not split across buffers. If any individual event exceeds that maximum configured size, the large event is sent in a buffer of its own, exceeding the configured value.

```
buffer_size=number_bytes
```

The default buffer size is 1024 bytes.

**flush_interval**

If events are being consolidated into very large buffers and there is not much logging activity, events can sit in memory for a long time before being forwarded to the remote server or being written to the cache file. The `flush_interval` option limits the time a process waits to fill a consolidation buffer. For example:

```
flush_interval={0|number_seconds}
```

The default flush interval is 20 seconds. A flush interval of 0 is not allowed. Specifying a value of 0 results in the buffer being flushed every 600 seconds.

**queue_size and hi_water**

The `queue_size` and `hi_water` values for a remote logging connection are similar to those specified for logging to a file.

```
queue_size={0|number_events}
hi_water={0|1|number}
```

The default value for queue size is 0 events, and the default value for the event queue high water mark is 100.

**compress**

Tivoli Access Manager events are principally text messages. To reduce network traffic use the `compress` option to compress buffers prior to transmission and expand on reception. For example:

```
compress={yes|no}
```

The default compress value is no.
error
If a send to a remote service fails, it is retried after waiting for the error retry timeout in seconds. If the retry also fails, the link will be marked down and this event and future events will be saved in the local event cache file until the remote service is rebound.

\texttt{error=seconds}

The default error retry timeout is 2 seconds.

path
The \texttt{path} option specifies the location of a cache file on the local host. The cache file name defaults to \texttt{/server.cache}, where \texttt{server} is the name of the remote server being logged to.

If the running process cannot establish communication with the remote server, or the link fails during operation, event recording switches to storing events in the specified file until the server again becomes available. When the server is available, events are drained from the disk cache and relayed to the remote server.

For example, suppose that the path value for \texttt{pdmgrd} on UNIX is as follows:

\texttt{path=/var/PolicyDirector/log/pdmgrd_remote.cache}

The directory portion of this path name must exist. The log file is created if it does not already exist. The size of this file is not bound, and it does not have any rollover capability. If a remote server is not accessible for sufficient time, you could run out of disk space.

rebind\_retry
If the remote server is unavailable, the log agent attempts to rebind to the server at this frequency in number of seconds.

\texttt{rebind\_retry=number\_seconds}

The default rebind retry timeout value is 300 seconds.

server
The remote logging services are offered by the \texttt{pdacld} program. Remote logging piggy-backs on the certificates set up for the authorization service as initialized by a call to \texttt{azn\_initialize()}. This server option nominates which hosts the \texttt{pdacld} process is bound to for event recording.

\texttt{server=hostname}

port
Use the \texttt{port} option to specify the port that the remote \texttt{pdacld} listens on for remote logging requests.

\texttt{port=pdacld\_port}

The default port value is 7136.

dn
To establish mutual authentication of the remote server, a distinguished name (DN) must be configured that can be checked against the name returned in the remote servers certificate.
The default value for the DN is a null string. Explicitly specifying an empty string or using the default value enables the logging client to request a remote server connection with any server listening.

Specifying a value for the DN limits successful connection to a specific server, such as:

```
dn="cn=ivacld/timelord.testnet.tivoli.com,o=policy\ director,c=us"
```

A DN must be specified as a string enclosed by double quotation marks.

---

**Event logging tasks**

The configuration key value pair, used for configuring Tivoli Access Manager server log files, are located in the `[aznapi-configuration]` stanza of each of these configuration files:

- `ivacld.conf` for the authorization server `pdacld`
- `ivmgrd.conf` for the policy server `pdmgrd`
- `pdmgrproxyd.conf` for the policy proxy server `pdmgrproxyd`
- `aznAPI.conf` for configured service plug-ins

The `aznAPI.conf` configuration file is provided with Tivoli Access Manager as a sample file. Developers of service plugins should provide the standard logging and auditing functions. Before implementing external authorization service plugins, read and thoroughly understand the concepts that are discussed in the *IBM Tivoli Access Manager for e-business Authorization C API Developer Reference*.

**Send event records to a log file**

Before you begin this task, review the information in "Configuration parameters for a file log agent" on page 203.

To configure Tivoli Access Manager to send event records to a log file:

1. Edit the appropriate server configuration file. Each server provides its own stanza entry values in its configuration file.
2. Locate the `[aznapi-configuration]` stanza.
3. Specify that the category is to send event records to a log file using the format `category:file`.
   
   For example, a category might be to audit authorization events (`audit.azn`):
   ```
   logcfg=audit.azn:file
   ```
4. Specify the log file location:
   ```
   path=fully_qualified_path
   ```
   
   The default directories are:
   
   **UNIX**  
   `/opt/PolicyDirector/log`
   
   **Windows**  
   `C:\Program Files\Tivoli\Policy Director\log`
   
   The default file name depends on the type of logging being performed, such as `audit.log`
5. Specify the log file ID:
   ```
   log_id=logid
   ```
Use the log_id option to set the log file identifier (ID) explicitly; otherwise, it is given a default value. If the path= option is specified, the default value is the configured path name. If path= is not specified, the log ID defaults to the domain component of the event category being captured. For example, logcfg = audit.azn:file implies log_id=audit

6. Specify the maximum log file size:
rollover_size= {<0|=0|>0}

By default, rollover_size=2000000.

The rollover size values are interpreted as:
- <0 — If less than zero, a new log file is created with each invocation of the process and every 24 hours from that instance.
- =0 — If equal to zero, no rollovers are performed, and the log file grows indefinitely. If a log file already exists, new data is appended to it.
- >0 — If greater than zero, a rollover is performed when a log file reaches the configured threshold value. If a log file already exists at startup, new data is appended to it.

7. Specify the maximum buffer size:
buffer_size= {0|number_kb}

By default, the buffer size for logging to a file is 0 bytes, which prevents buffering so each event is handled individually. If a number_kb value is specified, events are packed into buffers of that size before queueing to the file log agent.

Buffers consist of only an integral number of events; events are not split across buffers. If any individual event exceeds that maximum configured size, the large event is recorded in a buffer of its own, exceeding the configured value.

8. Specify the maximum number of events to queue in memory:
queue_size= {0|number_events}

By default, the queue size is 0. A zero queue size means that no limit is enforced on the growth of the unprocessed event queue. If the maximum number_events value is specified and reached and if a new event is ready to be placed on the queue, the requesting thread will be blocked until space is available in the queue.

9. Specify the event queue high water mark:
hi_water= {0|1|number}

By default, the event queue high water mark value is two-thirds of the maximum configured queue size.

If the maximum queue size is 0, the high water mark will be set to a default of 100. The transaction rates and the values of these options determine the maximum amount of memory that is consumed by enabling event logging to file.

If the event queue high water mark is set to 1, every event queued will be relayed to the log agent as soon as possible. This setting is not optimal.

10. Specify the frequency for flushing log file buffers:
flush_interval= {0|number_seconds}
11. Specify the file mode:
   
   ```
   mode={text|binary}
   ```


   Text mode is deprecated on UNIX platforms and has no effect.

12. Save and exit the configuration file.

This example shows the configuration parameters for a file log agent:

```
[aznapi-configuration]
logcfg=audit.azn:file path=/opt/PolicyDirector/log/audit.log,
flush_interval=20,rollover_size=2000000,log_id=audit,queue_size=200,
hi_water=100,buffer_size=2,mode=text
```

Note that tuning the buffer size with the queue size and the event queue high water mark can improve performance.

### Pipe event records to another program

Before you begin this task, review the information in “Configuration parameters for remote log agent” on page 208.

To configure Tivoli Access Manager to send event records to a remote server:

1. Edit the appropriate server configuration file. Each server provides its own stanza entry values in its configuration file.

2. Locate the [aznapi-configuration] stanza.

3. Specify the category using the format category:pipe. For example, a category might be to audit authorization events (audit):

   ```
   logcfg = audit:pipe
   ```

4. Specify that you want to pipe event records to another program (:pipe):

   ```
   logcfg = audit:pipe
   ```

5. Specify the program path to the location of the program, which receives the log output on standard input:

   ```
   path=fully_qualified_path
   ```

   There is no default value for the path name.

6. Specify the maximum number of events to queue in memory:

   ```
   queue_size={0|number_events}
   ```

   By default, the queue size is 0. A zero queue size means that no limit is enforced on the growth of the unprocessed event queue. If the maximum number_events value is specified and reached and if a new event is ready to be placed on the queue, the requesting thread will be blocked until space is available in the queue.

7. Specify the event queue high water mark:

   ```
   hi_water={0|1|number}
   ```

   By default, the event queue high water mark value is two-thirds of the maximum configured queue size.
If the maximum queue size is 0, the high water mark will be set to a default of 100. The transaction rates and the values of these options determine the maximum amount of memory that is consumed by enabling event logging to file.

If the event queue high water mark is set to 1, every event queued will be relayed to the log agent as soon as possible. This setting is not optimal.

8. Specify the frequency for flushing log file buffers:
   \texttt{flush\_interval=\{0\}<0\{number\_seconds\}}

A flush interval of 0 is not allowed. Specifying a value of zero results in the value 600 seconds being used.

If you specify a negative value ((<0)), the absolute value will be used as the asynchronous flush frequency, but a stream flush is also forced synchronously after every record is written.

To ensure stream buffers are flushed to disk regularly, the frequency with which the server asynchronously forces a flush of the file stream to disk is configurable using the \texttt{flush\_interval=number\_seconds} option.

9. Save and exit the configuration file.

This example pipes event records to a file named my_log_watcher:
\begin{verbatim}
[aznapi-configuration]
logcfg = audit:pipe
path=/opt/risk_analyst/bin/my_log_watcher,queue_size=0,hi_water=100,
flush_interval=300
\end{verbatim}

Send event records to a console

Before you begin this task, review the information in "Configuration parameters for console log agent" on page 202.

To configure Tivoli Access Manager to capture all audit output to standard out:

1. Edit the appropriate server configuration file. Each server provides its own stanza entry values in its configuration file.

2. Locate the [aznapi-configuration] stanza.

3. Specify the category for capturing events records. For example, a category might be to audit authorization events (audit):
   \texttt{logcfg = audit:destination}

4. Specify the capture of all event records to one of the following:
   a. To standard out using the format \texttt{category:stdout}.
      \texttt{logcfg = category:stdout}
   b. To standard error using the format
      \texttt{logcfg = category:stderr}

5. Save and exit the configuration file.

This example captures audit authorization events and sends them to a console using standard out:
\begin{verbatim}
[aznapi-configuration]
logcfg = audit:stdout
\end{verbatim}
Send event records to a remote server

Before you begin this task, review the information in "Configuration parameters for remote log agent" on page 208.

To configure Tivoli Access Manager to send event records to a remote server:

1. Edit the appropriate server configuration file. Each server provides its own stanza entry values in its configuration file.

2. Locate the [aznapi-configuration] stanza.

3. Specify that the category is to send event records to a remote server using the format category:remote.
   For example, a category might be to audit authorization events (audit):
   
   ```
   logcfg=audit:remote
   ```

4. Specify the maximum buffer size, which is maximum size message that the local program attempts to construct by combining smaller events into a large buffer:
   
   ```
   buffer_size={0|number_bytes}
   ```

   If a `number_bytes` value is specified, events are packed into buffers of that size before being relayed to the remote server. By default, the buffer size before relaying to the remote server is 1024 bytes.

   Buffers consist of only an integral number of events; events are not split across buffers. If any individual event exceeds that maximum configured size, the large event is recorded in a buffer of its own, exceeding the configured value.

5. Specify the frequency for flushing log file buffers:
   
   ```
   flush_interval={0|number_seconds}
   ```

   The `flush_interval` option limits the time a process waits to fill a consolidation buffer.

   By default, the flush interval value is 20 seconds. A flush interval of 0 is not allowed. Specifying a value of 0 results in the buffer being flushed every 600 seconds.

6. Specify the maximum number of events to queue:
   
   ```
   queue_size={0|number_events}
   ```

   By default, the queue size is 0. A zero queue size means that no limit is enforced on the growth of the unprocessed event queue. If the maximum value for `number_events` is specified and reached and if a new event is ready to be placed on the queue, the requesting thread will be blocked until space is available in the queue.

7. Specify the event queue high water mark:
   
   ```
   hi_water={0|1|number}
   ```

   By default, the event queue high water mark value is a `number` that represents two-thirds of the maximum configured queue size.

   If the maximum queue size is 0, the high water mark will be set to a default of 100. The transaction rates and the values of these options determine the maximum amount of memory that is consumed by enabling event logging to file.
If the event queue high water mark is set to 1, every event queued will be relayed to the log agent as soon as possible. This setting is not optimal.

8. Specify whether you want to compress buffers prior to transmission and expand on reception:
   
   `compress=(yes|no)`

   By default, the compress value is no to disable.

9. Specify the time to wait whenever a send to a remote service fails and an error occurs:
   
   `error=seconds`

   By default, the error retry timeout is 2 seconds.

10. Specify the cache file location:

    `path=fully_qualified_path`

    The file name is `server_name_remote.cache`. For example:

    `pdmgrd_remote.cache`

    The default directories are:

    **UNIX**
    `/opt/PolicyDirector/log`

    **Windows**
    `C:\Program Files\Tivoli\Policy Director\log\`

    The default file name depends on the type of logging being performed, such as `audit.log`

11. Specify the time between attempts to rebind (sign on):

    `rebind_retry=number_seconds`

    By default, the rebind retry timeout value is 300 seconds.

12. Specify the remote server host name:

    `server=hostname`

13. Specify the remote server port number:

    `port=pdacld_port`

    By default, the port number value is 7136.

14. Specify the remote server distinguished name to establish mutual authentication of the remote server:

    `dn="distinguished_name"`

    The default value for the DN is a null string. Explicitly specifying an empty string or using the default value enables the logging client request a remote server connection with any server listening.

    Specifying a value for the DN limits successful connection to a specific server, such as:

    `dn="cn=ivaclld/timelord.tivoli.com,o=policy director,c=us"`

    A DN must be specified as a string enclosed by double quotation marks.

15. Save and exit the configuration file.

This example sends event records to the remote server `timelord`:
Monitor log queue performance

The queuing profiles configured for the main propagation queue (as well as each file agent, remote agent, and pipe log agent) can be monitored by using the statistics interface.

Each queue is implemented by instantiating an EventQueue object that registers itself with the statistics subsystem using a category name constructed from the logging agent type and the string, pd.log.

The statistics of an event queue can be interrogated by using pdadmin server task commands. To establish what queues are implemented on a server, issue the server task server_name stats list command. A report similar to the following is returned:

```
pdadmin sec_master> server task ivacld-barra.surf.ap.tivoli.com stats list
pd.ras.stats.monitor
pd.log.EventPool.queue    // Main event propagation queue
pd.log.file.audit         // Audit log queue
```

To examine the statistics for a queue, enter on one line the stats get command:

```
pdadmin sec_master> server task ivacld-barra.surf.ap.tivoli.com stats get
pd.log.EventPool.queue
```

A report similar to the following is displayed:

```
dispatcher wakes on timeout(20) : 3617
dispatcher wakes by notify : 0
   notifies above highwater (100) : 0
   notifies below highwater : 0
spurious notifies : 0
total events processed : 24
average number of events handled per activation : 1
greatest number of events handled per activation : 7
blocks in queue requests : 0
```

The queue flush frequency is listed in parentheses after the word, timeout. The queue’s high water setting is listed in parentheses after the word, highwater.

The settings chosen for the various queue configuration options should attempt to balance the maximum amount of memory consumed between queue activations with the rate at which a particular log agent can consume events.

Optimally, you should set the queue high water mark such that the number of events processed during a queue activation fills a processing time slice. This setting avoids unnecessary thread context switching. Note however, that simply setting these options to high values is unlikely to be productive because event log processing must be done at some point and cannot be deferred indefinitely. Consuming large amounts of memory also has its own drawbacks.
Chapter 20. Logging of legacy auditing events

Tivoli Access Manager provides two methods of capturing audit events. One method is used to capture audit events for the current version of Tivoli Access Manager and the other method is used to capture audit events for legacy purposes. Use this method to capture audit events for versions prior to and including Version 3.8 of Tivoli Access Manager.

This chapter contains the following sections:

- “Overview of auditing”
- “Audit trail file contents” on page 220
- “Legacy configuration file stanza entries for auditing” on page 225
- “Logging and auditing tasks” on page 228

Overview of auditing

Auditing is defined as the logging of audit records. It includes the collection of data about system activities that affect the secure operation of the Tivoli Access Manager authorization process. Each Tivoli Access Manager server can capture audit events whenever any security-related auditable activity occurs.

Auditing uses the concepts of a record, an audit event, and an audit trail. Each audited activity is referred to as an audit event. The output of a specific server event is called a record. An audit trail is a collection of multiple records that document the server activity. All Tivoli Access Manager audit trail files are in ASCII format.

When configuring auditing, think of it as the source of the auditing events you want to gather. Audit trail files can capture authorization, authentication, and management events occurring on the Tivoli Access Manager servers. There are multiple sources for auditing events that you want to gather. You can collect either a combination or all of the different types of auditing events at the same time:

- authorization (azn)
- authentication (authn)
- management (mgmt)

Note: If you are using WebSEAL, there is an additional type of auditing: HTTP (http). See the IBM Tivoli Access Manager for e-business WebSEAL Administration Guide for an explanation of this type of auditing.

Only one type of destination is supported for the capture of auditing events:

- Audit trail file

Audit trail files are used by the Tivoli Access Manager servers to store records of server activity. Each Tivoli Access Manager server maintains its own audit trail file. Tivoli Access Manager audit trail files record events for the following servers:

- Policy server (pdmgrd)
- Authorization server (pdacld)
- Policy proxy server (pdmgrproxyd)
For applications developed by using the Authorization ADK, a configuration file is created by the resource manager application and populated as part of the configuration process. The configuration file contains the auditing configuration parameters. Each auditing configuration parameter is individually discussed under “Legacy configuration file stanza entries for auditing” on page 225.

### Audit trail file contents

This sections describes the contents of an audit trail file:

- “Examples of authorization audit records”
- “Examples of authentication audit records” on page 221
- “Examples of management audit records” on page 223

### Examples of authorization audit records

Authorization is the primary function of the Tivoli Access Manager servers. Authorization audit records can be captured when a target object in the Tivoli Access Manager authorization policy database (protected object space) has a POP attached to it that enables audit functionality.

See Chapter 9, “Protected object policy management,” on page 87.

You can configure auditing for a particular server by adding “azn” to the audit configuration list in the [aznapi-configuration] stanza of the server’s configuration file:

```
[aznapi-configuration]
auditcfg = azn
```

The following record is a sample audit record for getting user authorization credentials:

```
<event rev="1.2">
<date>2003-11-14-16:25:08.341+00:00I-----</date>
<outcome status="0">0</outcome>
<originator blade="pdmgrd"><component rev="1.2">azn</component>
<action>0</action>
<location>phaedrus</location>
</originator>
<accessor name=""><principal auth="IV_LDAP_V3.0">sec_master</principal>
</accessor>
<target resource="3"><object>IV_LDAP_V3.0:sec_master</object></target>
<data>azn_id_get_creds</data>
</event>
```

The following record is a sample audit record of a failed attempt at getting authorization credentials for the user fred:

```
<event rev="1.2">
<date>2003-11-14-16:25:08.341+00:00I-----</date>
<outcome status="268809242">1</outcome>
<originator blade="pdmgrd"><component rev="1.2">azn</component>
<action>0</action>
<location>phaedrus</location>
</originator>
<accessor name=""><principal auth="IV_LDAP_V3.0">fred</principal>
</accessor>
<target resource="0"><object>IV_LDAP_V3.0:sec_master</object></target>
<data>azn_id_get_creds</data>
</event>
```
The following is a sample of audit start and audit stop event records for the authorization (azn) component. The ellipses represents additional events that are logged between the beginning and end of auditing:

```
<event rev="1.2">
  <date>2003-11-14-16:25:08.341+00:00</date>
  <outcome status="0">0</outcome>
  <originator blade="pdmgrd"><component rev="1.2">azn</component></originator>
  <action>0</action>
  <location>phaedrus</location>
  <accessor>invalid</accessor>
  <target resource="5"><object></object></target>
  <data>
    <audit event="Start"/>
  </data>
</event>.
<event rev="1.2">
  <date>2003-12-10-20:58:16.584+00:00</date>
  <outcome status="0">0</outcome>
  <originator blade="pdmgrd"><component rev="1.2">azn</component></originator>
  <action>0</action>
  <location>phaedrus</location>
  <accessor>invalid</accessor>
  <target resource="5"><object></object></target>
  <data>
    <audit event="Stop"/>
  </data>
</event>
```

Examples of authentication audit records

Authentication of a principal is performed externally to Tivoli Access Manager during credential acquisition. Audit records can be captured by Tivoli Access Manager to record the success or failure of such authentication attempts.

You can configure auditing of authentication attempts by adding "authn" to the audit configuration list in the [aznapi-configuration] stanza of the server’s configuration file:

```
[aznapi-configuration]
auditcfg = authn
```

The following is a sample authentication event logged from WebSEAL for an unauthenticated user:

```
<event rev="1.2">
  <date>2003-11-14-23:04:26.630+00:00</date>
  <outcome status="0">0</outcome>
  <originator blade="websaald"><component rev="1.2">authn</component></originator>
  <action>0</action>
  <location>phaedrus</location>
  <accessor name=""/>
  <principal auth="invalid"></principal>
</accessor>
```
The following is a sample authentication event logged from WebSEAL for an authenticated user.

```
<event rev="1.2">
<date>2003-11-14-15:56:06.551+00:00</date>
<outcome status="0">0</outcome>
<originator blade="webseald"><component rev="1.2">authn</component>
<action>0</action>
<location>phaedrus</location>
</originator>
<target resource="7"><object></object></target>
<data></data>
</event>
```

The following is a sample authentication event that is caused by an expired password and that is logged from WebSEAL:

```
<event rev="1.2">
<date>2003-11-14-16:23:00.294+00:00</date>
<outcome status="320938188">0</outcome>
<originator blade="webseald"><component rev="1.2">authn</component>
<action>0</action>
<location>phaedrus</location>
</originator>
<target resource="7"><object></object></target>
<data></data>
</event>
```

Note that WebSEAL issues an outcome of 0 instead of 1 even though an expired password is encountered. It is considered successful because the correct password was entered.

The following is a sample WebSEAL authentication event that is caused by a bad password:

```
<event rev="1.2">
<date>2003-10-21-17:23:29.250-07:00</date>
<outcome status="320938184">1</outcome>
<originator blade="webseald"><component rev="1.2">authn</component>
<action>0</action>
<location>testsit</location>
</originator>
<target resource="7"><object></object></target>
<data>Password Failure: testuser</data>
</event>
```

The following is a sample successful change password event logged from WebSEAL. Note that the outcome status is 0.
The following is a sample authentication failure event because of too many invalid login attempts (three strikes policy) logged from WebSEAL.

To determine the reason for an audited event such as an account lock-out (three strikes policy), obtain the error code as shown in Table 10 on page 196. The error code is contained in the audit output in the <outcome status> tag:

```
<outcome status="320938290">0</outcome>
```

Invoke the `pdadmin errtext` command for the error code to receive a reason for the outcome. For example:

```
pdadmin>errtext 320938290
This account has been temporarily locked out due to too many failed login attempts
```

**Examples of management audit records**

The responsibilities of the policy server include maintaining the master authorization policy database. This database includes the description of the protected object space for the domain, access control lists (ACLs), protected object policies (POPs), and authorization rules and where these policies are attached to objects.

You can configure auditing of the policy server activity by adding "mgmt" to the audit configuration list in the [aznapi-configuration] stanza of the policy server's configuration file (ivmgrd.conf):

```
[aznapi-configuration]
auditcfg = mgmt
```

The following is a sample event record of the following `pdadmin` command:

```
pdadmin> pop modify pop1 set audit-level all
<event rev="1.2">
<date>2003-11-14-16:25:54.543+00:00I-----</date>
<outcome status="0">0</outcome>
<originator blade="ivmgrd"><component>mgmt</component>
<action>13702</action>
<location>phaedrus</location>
</originator>
<accessor name="">
<principal auth="password" domain="Default">testuser2</principal>
</accessor>
<target resource="7"><object></object></target>
<data>Account Lock-out: testuser2</data>
</event>
```
The following is a sample event record of the following **pdadmin** command issued by the sec_master user to create a new Tivoli Access Manager user with the surname of sngsouser1:

```
pdadmin> user create sngsouser1 "cn=sngsouser1,o=Tivoli,c=us" test user1 password12
```

```
<event rev="1.2">
<date>2003-07-02-23:35:05.723+00:00</date>
<outcome status="0">0</outcome>
<originator blade="pdmgrd"><component rev="1.2">mgmt</component>
<action>13401</action>
<location>c03comcrit1</location>
</originator>
<accessor name="">
<principal auth="IV_LDAP_V3.0">sec_master</principal>
</accessor>
</event>
```

The following is a sample event record of the following **pdadmin** command that disables the specified user (sngsouser1) account. The value of false disables the user account and relates to the account-valid no parameter. Likewise, the value of true enables the user account and relates to the account-valid yes parameter.

```
pdadmin> user modify sngsouser1 account-valid no
```

```
<event rev="1.2">
<date>2003-11-14-23:01:37.078+00:00</date>
<outcome status="0">0</outcome>
<originator blade="pdmgrd"><component>mgmt</component>
<action>13406</action>
<location>phaedrus</location>
</originator>
<accessor name="">
<principal auth="IV_LDAP_v3.0">dlucas</principal>
</accessor>
</event>
Legacy configuration file stanza entries for auditing

Stanza entries for configuring Tivoli Access Manager server audit trail files are located in the [aznapi-configuration] stanza of each of the server-name.conf files. Each resource manager names the configuration file. For applications developed by using the Authorization ADK, a configuration file is created by the resource manager application and populated as part of the configuration process. The configuration file contains the auditing configuration parameters.

The Tivoli Access Manager servers include:

<table>
<thead>
<tr>
<th>Server</th>
<th>server-name</th>
<th>Configuration File</th>
</tr>
</thead>
<tbody>
<tr>
<td>policy server</td>
<td>pdmgrd</td>
<td>ivmgrd.conf</td>
</tr>
<tr>
<td>policy proxy server</td>
<td>pdmgrproxyd</td>
<td>pdmgrproxyd.conf</td>
</tr>
<tr>
<td>authorization server</td>
<td>pdacld</td>
<td>ivacld.conf</td>
</tr>
</tbody>
</table>

For example:

For example, the following entries are needed for auditing:

```
[aznapi-configuration]
logaudit = {yes|no}
auditlog = fully_qualified_path
auditcfg = azn
auditcfg = authn
#auditcfg = mgmt
logsize = {-num|+num|byte_size of log prior to rollover}
logflush = num_seconds to flush event queue
```

**logaudit**

Audit trail recording is enabled on a server-by-server basis. Audit trail recording is enabled by setting the logaudit stanza entry value in the [aznapi-configuration] stanza of the configuration file for the specific server. By default auditing is disabled:

```
[aznapi-configuration]
logaudit = no
```

A value of yes enables auditing for that server. For example:

```
[aznapi-configuration]
logaudit = yes
```

**auditcfg**

Audit events are categorized by the server functionality that generates them. Some functionality is common across Tivoli Access Manager servers while other functionality is server-specific. Each type of server functionality is associated with an audit tag:

<table>
<thead>
<tr>
<th>Audit tag</th>
<th>Server functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>authn</td>
<td>Credential acquisition authentication auditing.</td>
</tr>
<tr>
<td>azn</td>
<td>Authorization event auditing.</td>
</tr>
<tr>
<td>mgmt</td>
<td>Management command auditing</td>
</tr>
</tbody>
</table>

You can configure each Tivoli Access Manager server to selectively capture audit events on a category-by-category basis. For example, the following configuration
captures only authentication events and disables the capture of all other events, including overriding any authorization auditing enabled in POP settings.

[aznapi-configuration]
auditcfg = authn

The following settings enable authentication and authorization auditing. All other audit categories are disabled by commenting out the line by starting the line with a pound sign (#):

[aznapi-configuration]
auditcfg = azn
auditcfg = authn
#auditcfg = mgmt

The http audit tag, which is used to capture WebSEAL HTTP events, is discussed in the IBM Tivoli Access Manager for e-business WebSEAL Administration Guide. The following setting enables WebSEAL HTTP auditing:

[aznapi-configuration]
auditcfg = http

By default, when auditing is enabled for a process with no configured audit tags, all auditable events are captured.

The following table indicates the auditing events (indicated by the audit tag) that can be captured for each specific Tivoli Access Manager server.

<table>
<thead>
<tr>
<th>Audit Tag</th>
<th>pdmgrd</th>
<th>pdacld</th>
<th>pdmgrproxyd</th>
<th>resource manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>authn</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>azn</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>mgmt</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**auditlog**

By default, the audit trail file for each server is called audit.log and is held in the specific server's log directory. The auditlog stanza entry value in each server's configuration file specifies the location of the audit trail file.

<table>
<thead>
<tr>
<th>Server</th>
<th>Log File Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>policy server (pdmgrd)</td>
<td>UNIX: auditlog = /var/PolicyDirector/audit/pdmgrd.log</td>
</tr>
<tr>
<td></td>
<td>Windows: auditlog = install_dir\audit\pdmgrd.log</td>
</tr>
<tr>
<td>policy proxy server (pdmgrproxyd)</td>
<td>UNIX: auditlog = /var/PolicyDirector/audit/pdmgrproxyd.log</td>
</tr>
<tr>
<td></td>
<td>Windows: auditlog = install_dir\audit\pdmgrproxyd.log</td>
</tr>
<tr>
<td>authorization server (pdacld)</td>
<td>UNIX: auditlog = /var/PolicyDirector/audit/pdacld.log</td>
</tr>
<tr>
<td></td>
<td>Windows: auditlog = install_dir\audit\pdacld.log</td>
</tr>
</tbody>
</table>
logsize

The value for the logsize stanza entry specifies the maximum size to which each of the audit trail files can grow and is initially configured with the following value (in bytes):

```
[aznapi-configuration]
logsize = 2000000
```

When an audit trail file reaches the specified value, known as its rollover threshold, the existing file is backed up to a file of the same name with an appended current date and timestamp. A new audit trail file is then started.

The various possible log size values are interpreted as follows:

- If the log size value is a negative number, meaning less than zero (< 0), a new audit trail file is created with each invocation of the auditing process and the logs are rolled over daily, regardless of the size.
- If the log size value is equal to zero (= 0), no rollover will be performed, and no audit trail will be created. If an audit trail file already exists, it grows indefinitely and the new data is appended to it.
- If the log size value is greater than zero (> 0), a rollover is performed when an audit trail file reaches the configured threshold value. The allowable range is from 1 byte to 2 megabytes. If an audit trail file already exists at startup, new data is appended to it.

logflush

Audit trail files are written to buffered data streams. If you are monitoring the audit trail files in real time, you might want to alter the frequency with which the server forces a flush of the audit trail file buffers.

By default, audit trail files are configured to be flushed every 20 seconds:

```
[aznapi-configuration]
logflush = 20
```

If you specify a negative value, the absolute value is used to determine when the audit trail files are flushed.

audit-attribute

You can specify the names of one or more access decision information (ADI) attributes to audit. An attribute can establish accountability by providing information to help identify potentially inappropriate access of assets. You can grant or deny access based on rules that are applied to attributes.

For example, the WebSEAL switch-user authentication feature provides a mechanism to allow certain users to impersonate another user. When switch-user is used, an authorization request is evaluated against an assumed identity rather than the actual identity of the user. It is desirable to allow administrators to capture the user’s actual identity.

For example, you can audit the names or descriptions of the Tivoli Access Manager policies (ACL, POP and authorization rule) that are applied to the object being accessed.

```
[aznapi-configuration]
audit-attribute = tagvalue_su-admin
```

There is no default value.
Logging and auditing tasks

The policy server manages the policy database or databases, and maintains location information about other Tivoli Access Manager servers in each domain. The policy server typically requires very little administration or configuration.

The configuration key value pair, used for configuring Tivoli Access Manager server audit trail files, are located in the [aznapi-configuration] stanza of each of these configuration files:

- `ivacld.conf` for the authorization server `pdacld`
- `ivmgrd.conf` for the policy server `pdmgrd`
- `pdmgrproxyd.conf` for the policy proxy server `pdmgrproxyd`
- `aznAPI.conf` for configured service plug-ins

The `aznAPI.conf` configuration file is provided with Tivoli Access Manager as a sample file. Developers of service plug-ins should provide the standard logging and auditing functions. Before implementing external authorization service plug-ins, read and thoroughly understand the concepts that are discussed in the *IBM Tivoli Access Manager for e-business Authorization C API Developer Reference*.

This section describes configuration tasks available to the administrator.

- “Enable auditing”
- “Specify which ADI attributes to audit” on page 229
- “Maintain audit log files” on page 229

Enable auditing

To configure Tivoli Access Manager server audit trail files:

1. Edit the appropriate server configuration file. Each server provides its own stanza entry values in its configuration file.
2. Locate the [aznapi-configuration] stanza.
3. Enable auditing by specifying yes or true:
   ```
   logaudit = {yes|true}
   ```

   By default, auditing is disabled. When enabled, the `auditcfg` and `auditlog` stanza entries are also required.
4. Select the component-specific type of audit records that you want to capture:
   ```
   auditcfg = azn
   #auditcfg = authn
   #auditcfg = mgmt
   ```

   By default, when auditing is enabled for a process with no configured audit tags, all auditable events are captured. Another stanza entry is `auditcfg=http`, which is used for components other than the Base, such as WebSEAL.

   Each server provides its own value in its configuration file.
5. Specify the name and location of the audit trail file for the local client:
   ```
   auditlog = fully_qualified_path
   ```

   If no location and name are supplied, auditing will not be performed.
Each server provides its own audit log setting in its corresponding configuration file. The following directories and file names are the default values for each server:

Windows
install_dir\audit\ivacld.log
install_dir\audit\pdmgrd.log
install_dir\audit\pdmgrproxyd.log

UNIX
/var/PolicyDirector/audit/pdmgrd.log
/var/PolicyDirector/audit/ivacld.log
/var/PolicyDirector/audit/pdmgrproxyd.log

6. Save and exit the configuration file.

This example shows the stanza entries needed to capture authorization events for the policy proxy server and redirecting the logging output to the pdmgrproxyd.log:

```
[aznapi-configuration]
logaudit = yes
auditlog = /var/PolicyDirector/audit/pdmgrproxyd.log
auditcfg = azn
#auditcfg = authn
#auditcfg = mgmt
```

### Specify which ADI attributes to audit

To specify the name of the ADI attribute that you to audit:

1. Edit the appropriate server configuration file.
2. Locate the [aznapi-configuration] stanza.
3. Add a stanza entry for each ADI attribute that you want to audit:

   ```
   audit-attribute = azn-attr1
   audit-attribute = azn-attr2
   ```

   There is no default value. The authorization API attribute (azn_attr) value represents an alphanumeric, case-insensitive string.

4. Save and exit the configuration file.

This example shows the configured stanza entries that are used to capture events for a WebSEAL attribute:

```
[aznapi-configuration]
logaudit = yes
auditlog = c:\myaudit.log
auditcfg = azn
audit-attribute = tagvalue_su-admin
```

### Maintain audit log files

To maintain audit log files:

1. Edit the appropriate server configuration file.
2. Locate the [aznapi-configuration] stanza.
3. Specify the maximum size in bytes of the audit log file before the rollover occurs:

   ```
   logsize = \{0\|neg_number_bytes\|number_bytes\}
   ```

   The allowable range is from 1 byte to 2 megabytes. The default value is 2000000.
Zero indicates that no rollover occurs and no rollover file is created. Any negative number indicates that the logs are rolled over daily, regardless of the size.

If the audit log file reaches its threshold, the original audit log file will be renamed and a new log file with the original name will be created.

4. Specify the frequency for flushing audit file buffers. This value will be the number of seconds allowed between log flushes. The valid range is from 1 second to 6 hours.

\[ \text{logflush} = \text{number}\_\text{seconds} \]

The default value is 20. If you specify a negative value, the absolute value is used to determine when the audit trail files are flushed.

5. Save and exit the configuration file.

This example shows the stanza entries needed to specify 2000000 bytes as the maximum size for the log file and 20 seconds between the flushing of log file buffers:

\[
\text{[aznapi-configuration]}
\text{logsize} = 2000000
\text{logflush} = 20
\]
Appendix A. Server configuration file reference

The operation of the Tivoli Access Manager servers is controlled through the use of configuration files. Each configuration file contains sections, called stanzas.

Configuration files

Server configuration files are ASCII text-based and contain stanza entries. Configuration files are processed only when the servers start. The following table lists the current Tivoli Access Manager configuration files.

<table>
<thead>
<tr>
<th>Configuration File</th>
<th>Purpose</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>“pd.conf” on page 235</td>
<td>Tivoli Access Manager runtime configuration</td>
<td>Used by the authorization server (pdacld)</td>
</tr>
<tr>
<td>“ivacld.conf” on page 233</td>
<td>Authorization server configuration</td>
<td>Tivoli Access Manager authorization server (pdacld)</td>
</tr>
<tr>
<td>“ivmgrd.conf” on page 234</td>
<td>Policy server configuration</td>
<td>Tivoli Access Manager policy server (pdmgrd)</td>
</tr>
<tr>
<td>“pdmgrproxyd.conf” on page 235</td>
<td>Policy proxy server configuration</td>
<td>Tivoli Access Manager policy proxy server (pdmgrproxyd)</td>
</tr>
<tr>
<td>“activedir_ldap.conf” on page 232</td>
<td>Microsoft Active Directory based user registry configuration</td>
<td>Microsoft Active Directory server</td>
</tr>
<tr>
<td>“ldap.conf” on page 234</td>
<td>LDAP-based user registry configuration</td>
<td>LDAP-based server</td>
</tr>
<tr>
<td>“activedir.conf” on page 232</td>
<td>Active Directory-based user registry configuration</td>
<td>Microsoft Active Directory server</td>
</tr>
<tr>
<td>“domino.conf” on page 233</td>
<td>Domino-based user registry configuration</td>
<td>IBM Lotus Domino server</td>
</tr>
</tbody>
</table>

If you did not change the installation directories when installing Tivoli Access Manager, the configuration files reside in the following default directories:

**UNIX** /opt/PolicyDirector/etc

**Windows** C:\Program Files\Tivoli\Policy Director\etc
**activedir.conf**

When you use the Microsoft Active Directory server as your user registry for Tivoli Access Manager, you must have the server configuration file `activedir.conf`. Use this configuration file to customize the operation of each Active Directory registry server.

**Note:** Active Directory is only supported on Microsoft Windows for the policy server.

Stanzas that can be included in this configuration file are:

- [uraf-registry]
- [meta-info]
- [ssl]
- [configuration-database]

The unconfiguration of the server using `activedir.conf` also queries information from this configuration file.

Also, you can set values for Active Directory stanza entries in the [uraf-registry] stanza in the `ivmgrd.conf` and `ivacld.conf` configuration files.

**activedir_ldap.conf**

When you use an LDAP client to retrieve data for the Active Directory user registry that the Tivoli Access Manager server is configured to, you must have the server configuration file `activedir_ldap.conf`. Use this configuration file to customize the operation of each Active Directory registry server.

For example, you might have multiple platforms where the policy server is configured to use the Active Directory user registry and other blades, such as WebSEAL on one platform, and the authorization server is configured to use the LDAP client to retrieve data from that Active Directory user registry on another platform.

Stanzas that can be included in this configuration file are:

- [uraf-registry]
**domino.conf**

When you use the Lotus Domino server as your user registry for Tivoli Access Manager, you must have the server configuration file `domino.conf`. Use this configuration file to customize the operation of each Domino registry server.

Stanzas that can be included in this configuration file are:

- [uraf-registry]

The unconfiguration of the server using `domino.conf` also queries information from this configuration file.

Also, you can set values for Domino stanza entries in the [uraf-registry] stanza in the `ivmgrd.conf` and `ivacld.conf` configuration files.

**ivacld.conf**

When you use the authorization server, `pdacld` for Tivoli Access Manager, you must have the server configuration file `ivacld.conf`. Use this configuration file to customize the operation of each authorization server.

Stanzas that can be included in this configuration file are:

- [meta-info]
- [ivacld]
- [ldap]
- [uraf-registry]
- [ssl]
- [manager]
- [authentication-mechanisms]
- [aznapi-configuration]
- [xmladi-attribute-definitions]
- [aznapi-entitlement-services]
- [aznapi-external-authzn-services]
- [aznapi-pac-services]
- [aznapi-cred-modification-services]
- [aznapi-admin-services]
- [configuration-database]

The unconfiguration of the server using `ivacld.conf` also queries information from this configuration file.
**ivmgrd.conf**

When you use the policy server pdmgrd for Tivoli Access Manager, you must have the server configuration file ivmgrd.conf. Use this configuration file to customize the operation of each policy server.

Stanzas that can be included in this configuration file are:

- [meta-info]
- [ivmgrd]
- [ldap]
- [uraf-registry]
- [ssl]
- [authentication-mechanisms]
- [aznapi-configuration]
- [xmladi-attribute-definitions]
- [aznapi-entitlement-services]
- [aznapi-pac-services]
- [aznapi-cred-modification-services]
- [aznapi-external-authzn-services]
- [delegated-admin]
- [configuration-database]
- [domains]
- [domain=domain_name]

The unconfiguration of the server using ivmgrd.conf also queries information from this configuration file.

**ldap.conf**

When you use LDAP as the user registry for Tivoli Access Manager, use the ldap.conf configuration file to customize the LDAP-based stanza entries.

Stanzas included in this configuration file include:

- [ldap]
- [meta-info]
- [ssl]

Note that the [ldap] stanza content is different when it appears in the activedir.conf and domino.conf configuration files.
**pd.conf**

For Tivoli Access Manager, you must have the `pd.conf` configuration file. Use this configuration file to automate server startup, to indicate whether the Tivoli Access Manager runtime package has been configured, and specify information about the user registry.

Stanza entries for automating server startup are located in the `[pdrte]` stanza of the `pd.conf` configuration file.

Stanzas that can be included in this configuration file are:

- `[meta-info]`
- `[pdrte]`
- `[ssl]`
- `[manager]`

The unconfiguration of the server using `pd.conf` also queries information from this configuration file.

**pdmgrproxyd.conf**

When you use the policy proxy server `pdmgrproxyd` for Tivoli Access Manager, you must have the server configuration file `pdmgrproxyd.conf`. Use this configuration file to customize the operation of each policy proxy server.

Stanzas that can be included in this configuration file are:

- `[meta-info]`
- `[pdmgrproxyd]`
- `[ldap]`
- `[uraf-registry]`
- `[ssl]`
- `[manager]`
- `[authentication-mechanisms]`
- `[aznapi-configuration]`
- `[xmladi-attribute-definitions]`
- `[aznapi-admin-services]`
- `[configuration-database]`

The unconfiguration of the server using `pdmgrproxyd.conf` also queries information from this configuration file.
**pdwpm.conf**

When you use the Tivoli Access Manager Web Portal Manager, you must have the configuration file `pdwpm.conf`. Use this configuration file to specify whether the change-password pages are to be displayed, and to indicate the location of the `pdwpm.properties` file.

You can also use this configuration file to indicate which type of authentication login method to use when you have WebSEAL junctioned.

Stanzas that can be included in this configuration file are:
- `[meta-info]`
- `[pdwpm]`

The unconfiguration of the server using `pdwpm.conf` also queries information from this configuration file.
Resource manager configuration files

Tivoli Access Manager provides a sample file containing the more common configuration file stanzas. Your documentation sources, when implementing your own plug-in or security-enhanced application, include the IBM Tivoli Access Manager for e-business Authorization C API Developer Reference or IBM Tivoli Access Manager for e-business Authorization Java Classes Developer Reference.

When creating your own security resource manager or extending the functions provided by Tivoli Access Manager, you can use the aznAPI.conf configuration file. This file is included as a sample with the authorization ADK package in the example/authzn/demo/cpp subdirectory.

Stanzas that can be included in this configuration file are:

- [aznapi-configuration]
- [xmladi-attribute-definitions]
- [ssl]
- [ldap]
- [uraf-registry]
- [aznapi-entitlement-services]
- [aznapi-pac-services]
- [aznapi-cred-modification-services]
- [aznapi-external-authzn-services]
- [aznapi-admin-services]
- [manager]
- [authentication-mechanisms]
Guidelines for configuring stanzas

These guidelines are provided to help you make changes to the Tivoli Access Manager configuration files. The guidelines are divided into these types:

- General guidelines
- Default values
- Strings
- Defined strings
- File names
- Integers
- Boolean values

For instructions, see “Change configuration settings” on page 144.

General guidelines

Use the following general guidelines when making changes to the configuration settings:

- There is no order dependency or location dependency for stanzas in any configuration file.
- Stanza entries are marked as required or optional. When an entry is required, the entry must contain a valid key and value.
- Do not change the names of the keys in the configuration files. Changing the name of the key might cause unpredictable results for the servers.
- Stanza entries and key names are case-sensitive. For example, useSSL and UseSSL are treated as different entries.
- Spaces are not allowed for names of keys.
- For the key value pair format of key = value, the spaces surrounding the equal sign (=) are not required, but they are recommended.
- Non-printable characters (such as tabs, carriage returns, and line feeds) that occur at the end of a stanza entry are ignored. Non-printable characters are ASCII characters with a decimal value less than 32.

Default values

Use the following guidelines when changing default configuration settings:

- Many values are created or modified only by using configuration programs. Do not manually edit these stanzas or values.
- Some values are filled in automatically during configuration. These values are needed for the initialization of the server after the configuration.
- The default values for a stanza entry might be different, depending on the server configuration. Some key value pairs are not applicable to certain servers and are omitted from the default configuration file for this server.
Strings
Some values accept a string value. When you manually edit the configuration file, use the following guidelines to change configuration settings that require a string:

- String values are expected to be characters that are part of the local code set.
- Additional or different restrictions on the set of allowable string characters might be imposed. For example, many strings are restricted to ASCII characters. Consult each stanza entry description for any restrictions.
- Double quotation marks are sometimes, but not always, required when you use spaces or more than one word for values. Refer to the descriptions or examples for each stanza entry when in doubt.
- The minimum and maximum lengths of user registry-related string values, if there are limits, are imposed by the underlying registry. For example, for Active Directory the maximum length is 256 alphanumeric characters.

Defined strings
Some values accept a string value, but the value must be one of a set of defined strings. When you manually edit the configuration file, make sure that the string value you type matches one of the valid defined strings values.

For example, the [aznapi-configuration] stanza section contains the following entry:

```
auditcfg = {azn|authn|mgmt}
```

The value for auditcfg is expected to be either azn, authn, or mgmt. Any other value is invalid and results in an error.

File names
Some values are file names. For each stanza entry that expects a file name as a value, the description of the stanza entry specifies which of the following constructs are valid:

- Filename
  No directory path included.
- Relative filename
  A directory path is allowed but not mandatory.
  These files typically are expected to be located relative to the location of a standard Tivoli Access Manager directory. The stanza entry for each relative path name lists the root directory to which the file name is relative.
- Fully qualified absolute path
  An absolute directory path is required.

Note: Some stanza entries allow more than one of the above choices.

The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows, file names cannot have these characters: a backward slash (\), a colon (:), a question mark (?), or double quotation marks (").
Integers

Many stanza entries expect the value for the entry to be expressed as an integer.

- Stanza entries that take an integer value expect integer values within a valid range. The range is described in terms of a minimum value and a maximum value.

  For example, in the [logging] stanza, the logflush stanza entry has a minimum value of 1 second and a maximum value of 600 seconds.

- For some entries, the integer value must be positive, and the minimum value is 1. For other entries, a minimum integer value of 0 is allowed.

  Use caution when setting an integer value to 0. For example, an integer value of 0 might disable the function that is controlled by that stanza entry. For example, in the [ivacld] stanza, the entry tcp-req-port = 0 disables the port number. Or, an integer value of 0 might indicate that the number is unlimited. For example, in the [ldap] stanza, the entry max-search-size = 0 means there is no limit to the maximum search size.

- For some entries requiring integer values, Tivoli Access Manager does not impose an upper limit for the maximum number allowed. For example, there is typically no maximum for timeout-related values, such as timeout = number in the [ldap] stanza.

  For this type of entry, the maximum number is limited only by the size of memory allocated for an integer data type. This number can vary, based on the type of operating system. For systems that allocate 4 bytes for an integer, this value is 2147483647.

  However, as the administrator, use a number that represents the value that is most logical for the value you are trying to set.

Boolean values

Many stanza entries represent a Boolean value. Tivoli Access Manager recognizes the Boolean values yes and no.

Some of the entries in the configuration files are read by other servers and utilities. For example, many entries in the [ldap] stanza are read by the LDAP client. Some of these other programs recognize additional Boolean characters:

- yes or true
- no or false

Anything other than yes|true, including a blank value, will be interpreted as no|false.

The recognized Boolean entries are listed for each stanza entry. Refer to the individual descriptions to determine when true or false are also recognized.
Stanzas

Stanza labels appear within brackets, such as: [stanza-name]. For example, the [ssl] stanza in the ivmgrd.conf configuration file defines the Secure Sockets Layer (SSL) configuration settings for the policy server. The [ldap] stanza defines configuration settings that are required by the policy server to communicate with an LDAP-based user registry.

Each stanza in a Tivoli Access Manager configuration file contains one or more key value pairs, which contain information that is expressed as a paired set of parameters. Each stanza entry has the following format:

\[key = value\]

You should not change the names of the keys in the configuration files. Changing the name of the key might cause unpredictable results in the servers. Note that spaces surrounding the equal sign (=) are not required but are recommended.

The initial installation of Tivoli Access Manager establishes many of the default values. Some values are static and will never change; other values can be modified to customize server functionality and performance.

In the following descriptions of each stanza, you will find a list of the valid stanza entries, consisting of key value pairs, for each stanza. The stanza entry includes a description of the default behavior for that entry.
[authentication-mechanisms] stanza

This stanza defines the libraries that are to be used for each form of authentication. Tivoli Access Manager supports only two forms (password and certificate authentication). Resource managers, such as WebSEAL, can support additional forms.

The configuration entries in this stanza are required by the server to communicate with a user registry. You can use either a User Registry Adapter Framework (URAF) registry (either Active Directory or Domino) or an LDAP registry library, depending on the type of user registry.

Because you can specify only one type of user registry, certain key value pairs in the [authentication-mechanisms] stanza are mutually exclusive.

```
passwd-ldap = fully_qualified_path
cert-ldap = fully_qualified_path
#passwd-uraf = fully_qualified_path
#cert-uraf = fully_qualified_path
```

In this example, the URAF registry items are commented out by using the pound sign (#); the LDAP-oriented stanza entries are not commented out.

The stanza entries for configuring the Tivoli Access Manager user registry are located in the [authentication-mechanism] stanza of each of these configuration files:
- `ivmgrd.conf` for the policy server `pdmgrd`
- `ivacld.conf` for the authorization server `pdacld`
- `pdmgrproxyd.conf` for the policy proxy server `pdmgrproxyd`
- Your resource managers’ configuration file

The `aznAPI.conf` configuration file is provided with Tivoli Access Manager as a sample file for creating your own resource manager configuration file. Developers of service plug-ins should provide the standard functions. Before implementing service plug-ins, read and thoroughly understand the concepts discussed in the *IBM Tivoli Access Manager for e-business Authorization C API Developer Reference*. 
### [authentication-mechanisms] stanza

**pwd-uraf = fully_qualified_path**

Location of the library to use for password authentication.

The _fully_qualified_path value represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows, file names cannot have these characters: a backward slash (\), a colon (:), a question mark (?), or double quotation marks ("). For UNIX, path and file names are case sensitive.

This stanza entry is required when you use a URAF registry as your user registry.

You can manually edit these values; no configuration utility is required.

Required only when the user registry is not LDAP.

Directory and files by platform include:

- AIX: /opt/PolicyDirector/lib/liburafauthn.a
- HP: /opt/PolicyDirector/lib/liburafauthn.sl
- Sun: /opt/PolicyDirector/lib/liburafauthn.so
- Linux: /opt/PolicyDirector/lib/liburafauthn.so
- Windows: <install_dir>\bin\urafauthn.dll

The default values are server-dependent.

Example for Windows: *pwd-uraf = C:\Program Files\Tivoli\Policy Director\bin\urafauthn.dll & -cfgfile [C:/pd/etc/server_name.conf]*

### cert-uraf = fully_qualified_path

Location of the library to use for certificate authentication.

The _fully_qualified_path value represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows, file names cannot have these characters: a backward slash (\), a colon (:), a question mark (?), or double quotation marks ("). For UNIX, path and file names are case sensitive.

This stanza entry is required when you use a URAF registry as the user registry.

You can manually edit these values; no configuration utility is required.

Required only when the user registry is not LDAP.

Directory and files by platform include:

- AIX: /opt/PolicyDirector/lib/liburafcertauthn.a
- HP: /opt/PolicyDirector/lib/liburafcertauthn.sl
- Solaris: /opt/PolicyDirector/lib/liburafcertauthn.so
- Linux: /opt/PolicyDirector/lib/liburafcertauthn.so
- Windows: <install_dir>\bin\urafcertauthn.dll

The default values are server-dependent.

Example for Windows: *cert-ldap = C:\Program Files\Tivoli\Policy Director\bin\certauthn.dll & -cfgfile [C:/pd/etc/server_name.conf]*
passwd-ldap = *fullyqualified_path*

Location of the library to use for LDAP password authentication.

The *fullyqualified_path* value represents an alphanumeric string. String values
are expected to be characters that are part of the local code set. The set of
characters permitted in a file name can be determined by the file system and by
the local code set. For Windows, file names cannot have these characters: a
backward slash (\), a colon (:), a question mark (?), or double quotation marks
(“). For UNIX, path and file names are case sensitive.

This stanza entry is required when you use LDAP as the user registry.

You can manually edit these values; no configuration utility is required.

Directory and files by platform include:

- AIX: `/opt/PolicyDirector/lib/libldapauthn.a`
- HP: `/opt/PolicyDirector/lib/libldapauthn.sl`
- Solaris: `/opt/PolicyDirector/lib/libldapauthn.so`
- Linux: `/opt/PolicyDirector/lib/libldapauthn.so`
- Windows: `install_dir\bin\ldapauthn.dll`

The default values are server-dependent.

Example for Solaris: `passwd-ldap = /opt/PolicyDirector/lib/libldapauthn.so
& -cfgfile [/opt/PolicyDirector/etc/server_name.conf]`

cert-ldap = *fullyqualified_path*

Location of the library to use for LDAP certificate authentication.

The *fullyqualified_path* value represents an alphanumeric string. String values
are expected to be characters that are part of the local code set. The set of
characters permitted in a file name can be determined by the file system and by
the local code set. For Windows, file names cannot have these characters: a
backward slash (\), a colon (:), a question mark (?), or double quotation marks
(“). For UNIX, path and file names are case sensitive.

This stanza entry is required when you use LDAP as the user registry.

You can manually edit these values; no configuration utility is required.

Directory and files by platform include:

- AIX: `/opt/PolicyDirector/lib/libcertauthn.a`
- HP: `/opt/PolicyDirector/lib/libcertauthn.sl`
- Solaris: `/opt/PolicyDirector/lib/libcertauthn.so`
- Linux: `/opt/PolicyDirector/lib/libcertauthn.so`
- Windows: `install_dir\bin\certauthn.dll`

The default values are server-dependent.

Example for Solaris: `cert-ldap = /opt/PolicyDirector/lib/libcertauthn.so
& -cfgfile [/opt/PolicyDirector/etc/server_name.conf]`
An administration service plug-in enables applications to perform application-specific administration tasks. The administration service plug-in is accessed by a calling application using one of the Tivoli Access Manager administration interfaces.

The calling application can be either an administrative utility such as the Tivoli Access Manager pdadmin command or the Tivoli Access Manager Web Portal Manager, or the calling application can be a custom-built application that uses the Tivoli Access Manager administration APIs.

The administration service maps the administration API calls to the corresponding administration service API calls, and carries out the requested action. Each administration service plug-in is a standalone module that is dynamically loaded into the authorization service.

The parameters for configuring Tivoli Access Manager administration service plug-ins are declared in the [aznapi-admin-services] stanza of these configuration files provided by Tivoli Access Manager:

- ivmgrd.conf for the policy server pdmgrd
- ivacld.conf for the authorization server pdacld
- pdmgrproxyd.conf for the policy proxy server pdmgrproxyd
- Your resource managers' configuration file for configured administration service plug-ins

The aznAPI.conf configuration file is provided with Tivoli Access Manager as a sample file for creating your own resource manager configuration file. Developers of service plug-ins should provide the standard functions. Before implementing service plug-ins, read and thoroughly understand the concepts discussed in the IBM Tivoli Access Manager for e-business Authorization C API Developer Reference.
## [aznapi-admin-services] stanza

### service-id = {short_name|path_to_dll} [-pobj protected_object_hierarchy_name] [& params]

Defines the authorization API service for functions that enable a plug-in to obtain the contents of a defined portion of the protected object hierarchy, or to enable a plug-in to define application-specific administration tasks that also return commands that perform those tasks. Each stanza entry defines different types of aznAPI service, and each entry is the same format where:

- **service-id**: Developer-specified identification (ID) of the administration service. An authorization API application can register more than one administration service plug-in, but each must have a unique service ID.

  - **{short_name|path_to_dll}**: The path to the dynamic link library (DLL) that contains the service executable code.

  - If the DLL resides in a directory that is normally searched by the system for DLLs (for example, /usr/lib on UNIX platforms and %PATH% on Windows NT), you do not need to specify the full path to the DLL, only the DLL name. If you want a platform-independent DLL name, so it can be loaded on any supported Tivoli Access Manager platform, provide a short form library name. The short name is prepended and appended with known library prefixes and suffixes for each platform, and each possibility is searched for in turn. For example, using a short form library name of `azn_ent_user`, the following names are automatically searched for on each platform:

    - NT: `azn_ent_user.dll`
    - AIX: `libazn_ent_user.so`, `libazn_ent_user.a`
    - Solaris: `libazn_ent_user.so`
    - HP/UX: `libazn_ent_user.sl`

- **protected_object_hierarchy_name**: The protected object hierarchy name is an optional parameter. This parameter refers to either the name of a protected object space (hierarchy) or simply a protected object. Protected object hierarchy names must be unique for each administration service plug-in within the scope of an authorization API application. Multiple authorization API application instances, however, can register to service the same protected object hierarchy names, which provides failover support for administration of an object space in the event that a particular authorization API application server fails.

- **params**: Optionally, the external authorization service can be passed additional initialization information in the form of arguments. The arguments must be preceded by the ampersand (for example, `%server fred`). The authorization service does not process the characters after the ampersand &. It passes these characters directly to the administration service plug-in. The service definition is discussed in more detail in the *IBM Tivoli Access Manager for e-business Authorization C API Developer Reference*.

This stanza entry is optional.

There is no default value.

### Example: AZN_ADMIN_SVC_TRACE = pdtraceadmin

<table>
<thead>
<tr>
<th>short_name</th>
<th>path_to_dll</th>
<th>protected_object_hierarchy_name</th>
<th>params</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
[aznapi-configuration] stanza

Tivoli Access Manager allows a highly flexible approach to authorization through the use of the authorization API. The standards-based authorization API allows applications to make calls to the centralized authorization service. Tivoli Access Manager provides built-in support of user name and password authentication as well through the authorization API.

Note that the configuration key value pair used for configuring Tivoli Access Manager server legacy audit trail files are located in the [aznapi-configuration] stanza of each of these configuration files:

- ivmgrd.conf for the policy server pdmgrd
- ivacld.conf for the authorization server pdacld
- pdmgrproxyd.conf for the policy proxy server pdmgrproxyd
- Your resource managers’ configuration files

Other stanza entries that apply to your resource managers’ configuration files are discussed in the IBM Tivoli Access Manager for e-business Authorization C API Developer Reference. Developers should read and thoroughly understand the concepts in this book to provide the required standard functions. A sample aznAPI.conf configuration file is provided with Tivoli Access Manager to use as a guide for creating your own resource manager configuration file.

### [aznapi-configuration] stanza

| mode = {local|remote} |
|-----------------------|
| Operating mode for the resource manager. This value cannot be changed after resource manager configuration. |
| **Note:** This stanza entry is set during configuration; do not change it. |
| Valid values include: |
| **local** | Resource manager uses a local policy cache. |
| **remote** | Resource manager uses a remote policy cache maintained by the authorization server (pdacld). |
| Some configuration attributes only apply to resource managers configured to use local mode. |
| This stanza entry is required. |
| Default value: local |
| Example: mode = remote |

<table>
<thead>
<tr>
<th>db-file = fully_qualified_path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name and location of the resource manager policy database cache file. This value must be specified, and each server provides its own value.</td>
</tr>
<tr>
<td>The <em>fully_qualified_path</em> value represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows, file names cannot have these characters: a backward slash (), a colon (:), a question mark (?), or double quotation marks (&quot;'). For UNIX, path and file names are case sensitive.</td>
</tr>
<tr>
<td>There is no default value.</td>
</tr>
<tr>
<td>Example for Windows: db-file = C:\pd\db\ivacld.db</td>
</tr>
<tr>
<td>Example showing relative path: db-file = ./authzn_demo.db</td>
</tr>
</tbody>
</table>
**cache-refresh-interval** = \{ disables|default\|number\_seconds\}

Poll interval (in seconds) between checks for updates to the master authorization policy database.

**Note:** The local cache is rebuilt only if an update is detected.

Valid values include:
- **disable** The interval value in seconds is not set.
- **default** The default value of 600 seconds is used.
- **number\_seconds**

The exact time interval that you set by specifying the number of seconds. The valid range is from 0 to the size of an unsigned integer, which is approximately 136 years.

This stanza entry is not used in the `ivmrd.conf` file because this server has its own stanza entries for specifying the master authorization policy database path.

This stanza entry is optional.

Default value: default

Example: `cache-refresh-interval` = 500

**listen-flags** = \{ enable|disable\}

Indication of whether to turn on or off the reception of policy cache update notifications.

Valid values include:
- **enable** Activates the notification listener.
- **disable** Deactivates the notification listener.

This stanza entry is optional.

Default value: disable

Example: `listen-flags` = enable

**policy-cache-size** = \texttt{size}\n
Maximum size of the in-memory policy cache. This size is configurable. The cache consists of policy and the relationships between policy and resources. The knowledge that a resource has no directly associated policy is also cached.

The maximum cache size should be relative to the number of policy objects defined and the number of resources protected as well as the available memory.

A reasonable algorithm to begin with is:

\[
\text{(number of policy objects \times 3) + (number of protected resources \times 3)}
\]

This value controls how much information is cached. A larger cache potentially improves the application performance, but uses additional memory as well.

Valid values include:
- **size**

Size is specified as the number of entries.

This stanza entry is optional.

Default value: 32768

Example: `policy-cache-size` = 32768
**azn-app-host = other_hostname**

Attribute that is used to specify the host name that the policy server should use when communicating with the resource manager.

For *other_hostname*, you can provide any valid internet host name. If this attribute is not specified, the default host name will be used. Examples of valid host names:

- mycomputer.city.company.com
- mycomputer

By default, this attribute is disabled. When disabled, the stanza entry is commented out by using a pound sign (#) at the beginning of the stanza entry in the configuration file. For example:

```text
#azn-app-host = libra
```

To enable this value, uncomment the entry in the configuration file by removing the pound sign (#). Be sure to include a host name value.

This stanza entry is optional.

There is no default value.

Example: `azn-app-host = libra.dallas.ibm.com`

**cred-attributes-entitlement-services = {short_name_entitlement_service|path_to_dll}**

Service that provides the ability to add external information to the user credential in the form of credential attributes and allows applications to use that information in making access decisions. These extended attributes are stored in the user registry.

This service can also work with attributes using an API call. A list of authorization API entitlement service IDs are queried by the `azn_id_get_creds()` interface to compile a list of attributes to be added to the user credential while the credential is being built.

Dynamic business entitlements (tag/value) no longer have to be loaded by using an authentication service module in WebSEAL nor are they limited just to the LDAP user registry.

A list of service IDs, which can be found within the `[aznapi-entitlement-services]` stanza, are queried to compile a list of attributes. The attributes are added to the user credential while the credential is being built. Each service ID is queried in the order it is declared in the list. The attribute returned is inserted into the credential attribute list of each credential that is built. For example:

```text
cred-attribute-entitlement-services = myEntSvcID
cred-attribute-entitlement-services = myOtherEntSvcID
```

**Note:** You cannot use this stanza entry to override read-only attributes in the credential attribute list that include the principal name, principal UUID, and others. The exception to this rule is for the `azn_cred_groups` attribute.

The *IBM Tivoli Access Manager for e-business Authorization C API Developer Reference* lists the read-only attributes, contains more information about this service, and explains why administrators who do not want this capability should ensure that the `azn_mod_rad` service is not loaded by the application.

This stanza entry is optional.

There is no default value.

Example: `cred-attribute-entitlement-services = myEntSvcID`
<table>
<thead>
<tr>
<th><strong>azn-server-name = server-hostname</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique name of the Tivoli Access Manager resource manager server, either <strong>pdmgrproxyd</strong>, <strong>pdacld</strong>, or <strong>pdmgrd</strong>, that is configured into the domain. The hyphen (–) character is required.</td>
</tr>
<tr>
<td><strong>Note:</strong> The host name is generated and set during configuration. Do not edit this stanza entry.</td>
</tr>
<tr>
<td>The <strong>server-hostname</strong> value represents an alphanumeric, case-insensitive string. String values are expected to be characters that are part of the local code set. The minimum and maximum lengths of the name are imposed by the underlying registry.</td>
</tr>
<tr>
<td>This stanza entry is optional.</td>
</tr>
<tr>
<td>The default value is server dependent.</td>
</tr>
<tr>
<td>Example: <strong>azn-server-name = ivacld-libra</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>pd-user-name = server_name/hostname</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tivoli Access Manager user account for the resource manager server, either <strong>pdmgrproxyd</strong>, <strong>pdacld</strong>, or <strong>pdmgrd</strong>, that is configured into the domain. The forward slash (/) character is required.</td>
</tr>
<tr>
<td><strong>Note:</strong> The server name/host name is generated and set during configuration. Do not edit this stanza entry.</td>
</tr>
<tr>
<td>The <strong>server_name/hostname</strong> value represents an alphanumeric, case-insensitive string. String values are expected to be characters that are part of the local code set. The minimum and maximum lengths of the name are imposed by the underlying registry.</td>
</tr>
<tr>
<td>Examples of valid host names:</td>
</tr>
<tr>
<td>• mycomputer.city.company.com</td>
</tr>
<tr>
<td>• mycomputer</td>
</tr>
<tr>
<td>This stanza entry is optional.</td>
</tr>
<tr>
<td>The default value is server dependent.</td>
</tr>
<tr>
<td>Example: <strong>pd-user-name = ivacld/libra</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>pd-user-pwd = server_password</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tivoli Access Manager user account password for the resource manager server, either <strong>pdmgrproxyd</strong>, <strong>pdacld</strong>, or <strong>pdmgrd</strong>, that is configured into the domain.</td>
</tr>
<tr>
<td><strong>Note:</strong> The server password is generated and set during configuration. Do not edit this stanza entry.</td>
</tr>
<tr>
<td>Password for the Tivoli Access Manager server.</td>
</tr>
<tr>
<td>This stanza entry is optional.</td>
</tr>
<tr>
<td>The default value is generated; do not change it.</td>
</tr>
<tr>
<td>Example: <strong>pd-user-pwd = zs77WoLSZn1rKr</strong></td>
</tr>
</tbody>
</table>
**permission-info-returned = \{attribute1 attribute2 ...\}**

Set of attributes that the caller wants to receive from the `azn_decision_access_allowed_ext()` function in the permission information attribute list. Before using this stanza entry tag and value, read and thoroughly understand the concept discussed in the *IBM Tivoli Access Manager for e-business Authorization C API Developer Reference*.

You can also define your own attributes. For example, you can set an attribute on an ACL by using the `acl modify set attribute` option of the `pdadmin` command.

When you add an attribute name to the list, the attribute can only be returned as permission information if it is applicable to the current decision call.

For a list of the strings recognized by the authorization engine, refer to *IBM Tivoli Access Manager for e-business Authorization C API Developer Reference*.

This stanza entry is optional.

Default value: No information is returned.

Example for returning permission information for all attributes in the list:

```
permission-info-returned = azn_perminfo_allAttrs
```

**resource-manager-provided-adi = prefix**

Prefix that the authorization engine uses to determine the set of missing access decision information (ADI) provided by the resource manager. To specify more than one prefix, add multiple stanza entries.

These entries must refer to existing entitlements services that were loaded using service entries in the `[aznapi-entitlement-services]` configuration stanza or initialization attribute. If an ADI is found to be missing during a rule evaluation, each service in this list is queried in the order defined.

The `prefix` stanza entry uses a string prefix for its value. For example, if you want to notify the authorization engine that any ADI beginning with `sales_customer_` be provided by the resource manager application, the stanza entry would be:

```
resource-manager-provided-adi = sales_customer_
```

Refer to **"resource-manager-provided-adi" on page 118** for more information about rule processing.

This stanza entry is optional.

There is no default value.

Example of multiple stanza entries:

```
resource-manager-provided-adi = sales_item_
resource-manager-provided-adi = sales_customer_
```
### dynamic-adi-entitlement-services = entitlement_service

Dynamic access decision information (ADI) retrieval entitlement service.

The *entitlement_service* is a string value for the container names of the required ADI. A list of configured authorization API entitlements service identifiers (IDs) are queried by the authorization rules engine when missing ADI is detected during an authorization rule evaluation.

When ADI is found to be missing during a rule evaluation, each service in this list is queried in the order defined in this entry. These stanza entries must refer to existing entitlements services.

The service ID (for example, bank_A_ADI) are loaded by using service entries in the entitlement service configuration [aznapi-entitlement-services] stanza or in an initialization attribute.

Refer to [“dynamic-adi-entitlement-services” on page 118](#) and the *IBM Tivoli Access Manager for e-business Authorization C API Developer Reference* for more information about rules processing and this service, respectively.

This stanza entry is optional.

There is no default value.

Example:

```
[aznapi-entitlement-services]
dynamic-adi-entitlement-services = bank_A_ADI
dynamic-adi-entitlement-services = bank_B_ADI
```

### input-adi-xml-prolog = prolog_attrs

Prolog to be added to the top of the XML document that is created using the Access Decision Information (ADI) needed to evaluate a Boolean authorization rule.

If a style sheet prolog is specified, that prolog will be imported into the empty style sheet. If no prolog is specified, a default prolog value will be used instead. All of the required prolog attributes are specified in the default prolog entries.

**Note:** If any of these attributes are changed or omitted from the entry, then the authorization client fails to start and returns an error.

The valid *prologAttrs* values are prolog attributes that are required by the authorization engine and include:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Refer to [“input-adi-xml-prolog and xsl-stylesheet-prolog” on page 119](#) for more information.

This stanza entry is optional.

Example: `input-adi-xml-prolog = <?xml version="1.0" encoding="UTF-8"?>`
**xsl-stylesheet-prolog = prolog_attrs**

The prolog to be added to the top of the XSL stylesheet that is created using the XSL text that defines a boolean authorization rule.

The valid `prolog_attrs` values are prolog attributes that are required by the authorization engine.

If a style sheet prolog is specified, that prolog will be imported into the empty style sheet. If no prolog is specified, a default prolog value will be used instead. All of the required prolog attributes are specified in the default prolog entries.

When not specified, the default XSL stylesheet prolog is:

```xml
<!-- Required for XSLT language -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <!-- Required to constrain output of rule evaluation -->
  <xsl:output method="text" omit-xml-declaration="yes"
            encoding="UTF-8" indent="no"/>
  <!-- Need this to ensure default text node printing is off -->
  <xsl:template match="text()"></xsl:template>
</xsl:stylesheet>
```

**Note:** If any of the required prolog attributes are changed or omitted from the entry, then the authorization client fails to start and returns an error.

Use caution when changing this setting. Refer to ["input-adi-xml-prolog and xsl-stylesheet-prolog" on page 119](#) for more information.

This stanza entry is optional.

See ["Defining an XML namespace" on page 108](#) for a complete explanation of the name space example.
audit-attribute = azn-attr

Name of the access decision information (ADI) attribute to audit. An attribute can establish accountability by providing information to help identify potentially inappropriate access of assets. You can grant or deny access based on rules that are applied to attributes.

For example, the WebSEAL switch-user authentication feature provides a mechanism to allow certain users to impersonate another user. When switch-user is used, an authorization request is evaluated against an assumed identity rather than the actual identity of the user. It is desirable to allow administrators to capture the user’s actual identity.

You can audit the names or descriptions of the Tivoli Access Manager policies (ACL, POP and authorization rule) that are applied to the object being accessed.

The authorization API attribute (azn_attr) value represents an alphanumeric, case-insensitive string. String values are expected to be characters that are part of the local code set.

This stanza entry is optional.

There is no default value.

Example for WebSEAL:

logaudit = yes
auditlog = audit.log
auditcfg = azn
audit-attribute = tagvalue_su-admin

logcfg =
category:[stdout|stderr|file|pipe|remote][param=value][param=value]]

Enables logging and auditing for the application. Category, destination, and other parameters are used to capture Tivoli Access Manager auditing and logging events.

Each server provides its own event logging setting in its corresponding configuration file.

Valid values include:

<table>
<thead>
<tr>
<th>category</th>
<th>log-agent</th>
<th>param=value</th>
</tr>
</thead>
<tbody>
<tr>
<td>log-agent</td>
<td>Category of auditing event. Also indicates that the destination where log-agent is one of stdout, stderr, file, pipe, or remote. For example: audit.azn:file</td>
<td></td>
</tr>
<tr>
<td>param=value</td>
<td>Allowable parameters. The parameters vary, depending on the category, the destination of events, and the type of auditing you want to perform. For example: hi_water = number</td>
<td></td>
</tr>
</tbody>
</table>

Refer to Chapter 19, “Capturing logging and auditing events,” on page 197 for information about the log agents and the configuration parameters.

This stanza entry is optional.

Remove the pound signs (#) at the beginning of the configuration file lines to enable authentication or authorization auditing (or both) for the application.

Default value:

#logcfg = audit.azn:file path=audit.log,flush_interval=20, log_id=audit_log
#logcfg = audit.authn:file log_id=audit_log
| logaudit = {yes|true|no|false} |
|----------------|
| Indication of whether to perform legacy auditing. |
| Valid values include: |
| yes|true Legacy auditing is enabled (turned on). |
| no|false Legacy auditing is disabled (turned off). |
| Default value: no. However, the default value is server-dependent. If logaudit = yes, you must also set the auditlog and auditcfg. |
| Example for Windows: |
| logaudit = yes |
| auditlog = C:\pd\audit\pdacld.log |
| auditcfg = azn |

| auditcfg = {azn|authn|mgmt|http} |
|----------------|
| Capture of legacy audit logging configuration events. |
| To enable component-specific audit records, add or remove the appropriate definition. Each server provides its own value in its configuration file. For example, default behavior for aznAPI.conf is azn; default behavior for pdacld.conf is also azn. |
| More than one auditcfg value can be used at the same time. Comment out the stanza entries that do not apply. To comment out a stanza entry, start the entry with a pound sign (#). For example: |
| auditcfg = azn |
| auditcfg = authn |
| #auditcfg = mgmt |
| Valid values for the type of event to be audited include: |
| azn Captures authorization events. |
| authn Captures authentication events. |
| mgmt Captures policy server events. |
| http Captures WebSEAL HTTP events. Refer to the IBM Tivoli Access Manager for e-business WebSEAL Administration Guide for more information about this value. |
| This stanza entry is required if logaudit = yes. You must also set auditlog. |
| There is no default value. The value depends on the type of event you want to audit: |
| Example for Windows to audit authorization events for the authorization server pdacld: |
| logaudit = yes |
| auditlog = C:\pd\audit\pdacld.log |
| auditcfg = azn |
### auditlog = fullyQualifiedPath

Name and location of the legacy audit trail file for the local client. If no location and name are supplied, auditing will not be performed. Each server provides its own audit log setting in its corresponding configuration file.

The `fullyQualifiedPath` value represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows, file names cannot have these characters: a backward slash (\), a colon (:), a question mark (?), or double quotation marks ("). For UNIX, path and file names are case sensitive.

This stanza entry is optional. If `logaudit` = yes, you must also set `auditcfg`.

The log file name is server-dependent.

The default installation location for Windows: `C:\pd_install_path\audit\server_name.log`

The default installation location for UNIX: `auditlog = /var/PolicyDirector/audit/server_name.log`

Example for Windows for the policy server `pdmgrd`:

```plaintext
logaudit = yes
auditlog = C:\pd\audit\pdmgrd.log
auditcfg = azn
```

### logsize = {0 | neg_number_bytes | number_bytes}

Log file rollover threshold (in bytes) for legacy audit logs. If the audit log file reaches this threshold, the original audit log file will be renamed and a new log file with the original name will be created.

Valid values include:

- `0` Zero indicates that no rollover occurs and no rollover file is created.
- `neg_number_bytes` Any negative number indicates that the logs are rolled over daily, regardless of the size.
- `number_bytes` The maximum size (in bytes) of the audit log file before the rollover occurs. The allowable range is from 1 byte to 2 megabytes.

This stanza entry is optional.

The default value is server-dependent.

Example: `logsize = 2000000`. 256 IBM Tivoli Access Manager: Base Administration Guide
**logflush = number_seconds**

<table>
<thead>
<tr>
<th>Time interval (in seconds) between log flushes of log file buffers for legacy audit logs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid values include:</td>
</tr>
<tr>
<td><em>number_seconds</em></td>
</tr>
<tr>
<td>The number of seconds allowed between log flushes. The valid range is from 1 second to 6 hours.</td>
</tr>
<tr>
<td>If you specify a negative value, the absolute value is used to determine when the audit trail files are flushed.</td>
</tr>
<tr>
<td>This stanza entry is optional.</td>
</tr>
<tr>
<td>The default value is server-dependent.</td>
</tr>
<tr>
<td>Example: <em>logflush = 20.</em></td>
</tr>
</tbody>
</table>

Appendix A. Server configuration file reference 257
[aznapi-cred-modification-services] stanza

A credential modification service plug-in enables authorization API applications to perform modifications on a Tivoli Access Manager credential. The credentials modification service can then return this modified credential for use by the calling application. Applications can use this service to add additional information to a user’s credential. For example, this additional information could include the user’s credit card number and the user’s credit limit. Each credential modification service plug-in is a standalone module that is dynamically loaded into the authorization service.

The parameters for configuring Tivoli Access Manager credential modification service plug-ins are declared in the [aznapi-cred-modification-services] stanza of each of the configuration files provided with Tivoli Access Manager:

- ivmgrd.conf for the policy server pdmgrd
- ivacld.conf for the authorization server pdacld
- pdmgrproxyd.conf for the policy proxy server pdmgrproxyd
- Your resource managers’ configuration file for configured credentials modification service plug-ins

The aznAPI.conf configuration file is provided with Tivoli Access Manager as a sample file for creating your own resource manager configuration file. Developers of service plug-ins should provide the standard functions. Before implementing service plug-ins, read and thoroughly understand the concepts discussed in the IBM Tivoli Access Manager for e-business Authorization C API Developer Reference.
## [aznapi-cred-modification-services] stanza

<table>
<thead>
<tr>
<th>service-id = short_name</th>
<th>path_to_dll [ &amp; params ... ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defines the authorization API service for the credentials attribute list modification service. Each stanza entry defines different types of aznAPI service, and each entry is the same format where:</td>
<td></td>
</tr>
<tr>
<td>service-id</td>
<td></td>
</tr>
<tr>
<td>Developer-specified identification (ID) of the credential modification service. The service ID string must be unique.</td>
<td></td>
</tr>
<tr>
<td>{short_name}</td>
<td>path_to_dll}</td>
</tr>
<tr>
<td>The path to the dynamic link library (DLL) that contains the service executable code.</td>
<td></td>
</tr>
<tr>
<td>If the DLL resides in a directory that is normally searched by the system for DLLs (for example, /usr/lib on UNIX platforms and %PATH% on Windows NT), you do not need to specify the full path to the DLL, only the DLL name. If you want a platform-independent DLL name, so it can be loaded on any supported Tivoli Access Manager platform, provide a short form library name. The short name is appended with known library prefixes and suffixes for each platform, and each possibility is searched for in turn. For example, using a short form library name of azn_ent_user, the following names are automatically searched for on each platform:</td>
<td></td>
</tr>
<tr>
<td>NT: azn_ent_user.dll</td>
<td></td>
</tr>
<tr>
<td>AIX: libazn_ent_user.so, libazn_ent_user.a</td>
<td></td>
</tr>
<tr>
<td>Solaris: libazn_ent_user.so</td>
<td></td>
</tr>
<tr>
<td>HP/UX: libazn_ent_user.sl</td>
<td></td>
</tr>
<tr>
<td>params Optionally, you can specify parameters to pass to the service when it is initialized by the aznAPI. Parameters are considered to be all data following the ampersand (&amp;) symbol in the string. The service definition is discussed in more detail in the IBM Tivoli Access Manager for e-business Authorization C API Developer Reference.</td>
<td></td>
</tr>
<tr>
<td>This stanza entry is optional.</td>
<td></td>
</tr>
<tr>
<td>There is no default value.</td>
<td></td>
</tr>
<tr>
<td>Example: AZN_MOD_SVC_RAD_2AB = azn_mod_rad</td>
<td></td>
</tr>
</tbody>
</table>

Appendix A. Server configuration file reference 259
[aznapi-entitlement-services] stanza

An entitlement services plug-in enables authorization API applications to retrieve the entitlements for a user from an entitlements repository. Each entitlement services plug-in is a standalone module that is dynamically loaded into the authorization service.

The stanza entries for configuring Tivoli Access Manager entitlement services plug-ins are declared in the [aznapi-entitlement-services] stanza of each of these configuration files provided by Tivoli Access Manager:
• ivmgrd.conf for the policy server pdmgrd
• ivacld.conf for the authorization server pdacld
• pdmgrproxyd.conf for the policy proxy server pdmgrproxyd
• Your resource managers’ configuration file for configured entitlement services plug-ins

The aznAPI.conf configuration file is provided with Tivoli Access Manager as a sample file for creating your own resource manager configuration file. Developers of service plug-ins should provide the standard functions. Before implementing service plug-ins, read and thoroughly understand the concepts discussed in the IBM Tivoli Access Manager for e-business Authorization C API Developer Reference.
## [aznapi-entitlement-services] stanza

| service-id | (short_name|path_to_dll) | [ & params ... ] |
|------------|--------------|------------------|
|            |              |                  |

Defines the authorization API service for the protected objects entitlements service. Each stanza entry defines different types of aznAPI service, and each entry is of the same format where:

- **service-id**
  - Developer-specified identification (ID) by which the service can be identified by the aznAPI client. The service ID string must be unique.

- **short_name|path_to_dll**
  - The path to the dynamic link library (DLL) that contains the service executable code.
  - If the DLL resides in a directory that is normally searched by the system for DLLs (for example, /usr/lib on UNIX platforms and %PATH% on Windows NT), you do not need to specify the full path to the DLL, only the DLL name. If you want a platform-independent DLL name, so it can be loaded on any supported Tivoli Access Manager platform, provide a short form library name. The short name is appended with known library prefixes and suffixes for each platform, and each possibility is searched for in turn. For example, using a short form library name of azn_ent_user, the following names are automatically searched for on each platform:
    - NT: azn_ent_user.dll
    - AIX: libazn_ent_user.so, libazn_ent_user.a
    - Solaris: libazn_ent_user.so
    - HP/UX: libazn_ent_user.sl

- **params**
  - Optionally, you can specify one or more parameters to pass to the service when it is initialized by the aznAPI. Parameters are considered to be all data following the ampersand (&) symbol in the string. The service definition is discussed in more detail in the *IBM Tivoli Access Manager for e-business Authorization C API Developer Reference*.

This stanza entry is optional.

There is no default value.

Example: credattrs_ent_svc = azn_ent_cred_attrs
[aznapi-external-authzn-services] stanza

An external authorization service plug-in is an optional extension of the Tivoli Access Manager authorization service that allows you to impose additional authorization controls and conditions. You can use an external authorization service plug-in to force authorization decisions to be made based on application-specific criteria that are not known to the Tivoli Access Manager authorization service. Each external authorization service plug-in is a standalone module that is dynamically loaded into the authorization service.

The parameters for configuring Tivoli Access Manager external authorization service plug-ins are declared in the [aznapi-external-authzn-services] stanza of this configuration file provided by Tivoli Access Manager:

- ivmgrd.conf for the policy server pdmgrd
- ivacld.conf for the authorization server pdacld
- Your resource managers’ configuration file for configured external authorization service plug-ins

The aznAPI.conf configuration file is provided with Tivoli Access Manager as a sample file for creating your own resource manager configuration file. Developers of service plug-ins should provide the standard functions. Before implementing service plug-ins, read and thoroughly understand the concepts discussed in the IBM Tivoli Access Manager for e-business Authorization C API Developer Reference.
### aznapi-external-authzn-services stanza

```markdown
policy-trigger = {short_name|path_to_dll} [ & params ... ]
```

Defines the authorization API service for external authorization service definitions that force authorization decisions to be made based on application-specific criteria. Each stanza entry defines different types of aznAPI service, and each entry is the same format where:

**policy-trigger**

The policy trigger is the means by which the external authorization service is invoked by the authorization engine. It is one of either a service ID or an access control list (ACL) action string. For example, it can be `my_service_1` or `Trx`. If the service is defined with a service ID, the service ID will be used as an extended attribute on a POP policy that triggers the external authorization service whenever an object has this POP attached to it. If the service is defined using an ACL action string, the service will be invoked whenever this ACL action mask is requested as part of an authorization decision.

The policy-trigger can be any string that is recognized as a valid key name. The policy-trigger string is case sensitive for action set definitions because the actions themselves are case sensitive. However, the policy-trigger is case insensitive if the trigger is a POP attribute.

**[short_name|path_to_dll]**

The path to the dynamic link library (DLL) that contains the service executable code.

If the DLL resides in a directory that is normally searched by the system for DLLs (for example, `/usr/lib` on UNIX platforms and `%PATH%` on Windows NT), you do not need to specify the full path to the DLL, only the DLL name. If you want a platform-independent DLL name, so it can be loaded on any supported Tivoli Access Manager platform, provide a short form library name. The short name is appended with known library prefixes and suffixes for each platform, and each possibility is searched for in turn. For example, using a short form library name of `azn_ent_user`, the following names are automatically searched for on each platform:

- **NT**: `azn_ent_user.dll`
- **AIX**: `libazn_ent_user.so, libazn_ent_user.a`
- **Solaris**: `libazn_ent_user.so`
- **HP/UX**: `libazn_ent_user.sl`

**[-weight number]**

A weighting that is assigned in the access decision process to the particular external authorization service. The weight parameter is an unsigned `size_t` value and is optional. The value signifies the weight that any decision returned by this external authorization service should be given in the entire decision process. The default value is 101.

**params**

Optionally, the external authorization service can be passed additional initialization information in the form of arguments. The arguments must be preceded by the ampersand (for example, `& -server fred`). The service definition is discussed in more detail in the *IBM Tivoli Access Manager for e-business Authorization C API Developer Reference*.

This stanza entry is optional.

There is no default value.
[aznapi-pac-services] stanza

A PAC services plug-in gives authorization API applications the ability to move Tivoli Access Manager credentials back and forth between the native Tivoli Access Manager credentials format and an alternate format called *privilege attribute certificate* (PAC). Each PAC services plug-in is a standalone module that is dynamically loaded into the authorization service.

Identity information can be obtained from a PAC. Applications can convert user credentials to PACs for use within other authorization domains. Applications can then pass the PACs to a server in another authorization domain and perform an operation.

The stanza entries for configuring Tivoli Access Manager PAC services plug-ins are declared in the [aznapi-pac-services] stanza of each of these configuration files provided by Tivoli Access Manager:

- Your resource managers' configuration file for configured PAC services plug-ins

The aznAPI.conf configuration file is provided with Tivoli Access Manager as a sample file for creating your own resource manager configuration file. Developers of service plug-ins should provide the standard functions. Before implementing service plug-ins, read and thoroughly understand the concepts discussed in the *IBM Tivoli Access Manager for e-business Authorization C API Developer Reference*. 
### [aznapi-pac-services] stanza

| service-id = {short_name|path_to_dll} [ & params ... ] |
|--------------------------------------------------------|
| **service-id** |
| Developer-specified identification (ID) of the PAC service that produces the PAC. The service ID string must be unique. |
| **{short_name|path_to_dll}** |
| The path to the dynamic link library (DLL) that contains the service executable code. |
| If the DLL resides in a directory that is normally searched by the system for DLLs (for example, /usr/lib on UNIX platforms and %PATH% on Windows NT), you do not need to specify the full path to the DLL, only the DLL name. If you want a platform-independent DLL name, so it can be loaded on any supported Tivoli Access Manager platform, provide a short form library name. The short name is appended with known library prefixes and suffixes for each platform, and each possibility is searched for in turn. For example, using a short form library name of azn_ent_user, the following names are automatically searched for on each platform: |
| **NT:** azn_ent_user.dll |
| **AIX:** libazn_ent_user.so, libazn_ent_user.a |
| **Solaris:** libazn_ent_user.so |
| **HP/UX:** libazn_ent_user.sl |
| **params** |
| Optionally, you can specify parameters to pass to the service when it is initialized by the aznAPI. Parameters are considered to be all data following the ampersand (&) symbol in the string. The service definition is discussed in more detail in the *IBM Tivoli Access Manager for e-business Authorization C API Developer Reference*. |
| This stanza entry is optional. |
| There is no default value. |

---

Appendix A. Server configuration file reference 265
[configuration-database] stanza

The stanza entry defines the name and location of the Tivoli Access Manager obfuscated password configuration file. Tivoli Access Manager creates a new configuration file containing all the obfuscated entries. For example, All bind (log in) passwords are obfuscated and placed in the configuration file. Both the existing configuration file and the obfuscated configuration file have the same file name, except that .obf is appended to the file name (for example, ivmgrd.conf.obf).

In addition, Tivoli Access Manager creates the [configuration-database] stanza, as needed, whenever an obfuscated entry is automatically added to the obfuscated configuration file. This stanza has a stanza entry that points to the name and location of the obfuscated configuration file. The [configuration-database] stanza can be located in every configuration file, including the pd.conf configuration file, if an obfuscated value is added to the file.

You should never edit the entry in the [configuration-database] stanza. The one exception might be if the file is to be moved permanently to a different location. This scenario is the only circumstance where the file name and location should be modified. Remember that whenever the configuration file is moved to a different location, you must move the obfuscated file also.

<table>
<thead>
<tr>
<th>[configuration-database] stanza</th>
</tr>
</thead>
<tbody>
<tr>
<td>file = fullyQualifiedPath</td>
</tr>
</tbody>
</table>

File name and location where the obfuscated configuration file information resides.

Note: The obfuscated password is generated and set by the configuration utility. Do not edit this stanza entry.

The name of the obfuscated configuration file is the same name as the related configuration file name. The file type can be anything, but the extension is usually .conf.obf. For example, the obfuscated configuration file for ldap.conf is ldap.conf.obf.

The fullyQualifiedPath value represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows, file names cannot have these characters: a backward slash (\), a colon (:), a question mark (?), or double quotation marks ("). For UNIX, path and file names are case sensitive.

This stanza entry is required only if, during configuration, passwords have been obfuscated. Default installation location for UNIX:

/opt/PolicyDirector/etc/server_name.conf.obf

Default installation location for Windows: c:\Program files\tivoli\Policy Director\etc\server_name.conf.obf

Example for Windows when Microsoft Active Directory is the user registry:

C:\Program Files\Tivoli\Policy Director\etc\activeDir.conf.obf
[delegated-admin] stanza

The Tivoli Access Manager configuration can require that the user be authorized to view each group that is returned in the group list. Or, the user can be authorized to return the list without authorizing first.

For delegated administration, you should use one type of interface throughout the entire process for optimal results. Use either the Web Portal Manager or the pdadmin command. This stanza relates only to the pdadmin command.

The stanza entries for turning on or off the setting for authorization checks for delegated management of groups and users are located in the [delegated-admin] stanza of this configuration file:

• ivmgrd.conf for the policy server pdmgrd

<table>
<thead>
<tr>
<th>[delegated-admin] stanza</th>
</tr>
</thead>
<tbody>
<tr>
<td>authorize-group-list = {yes</td>
</tr>
</tbody>
</table>

Indication of whether authorization checks on the pdadmin group list and group list-dn commands should be made.

This keyword is provided as a performance feature.

Valid values include:

- yes Enables authorization checks.
- no Disables authorization checks.

This stanza entry is optional.

Default value: no

Example: authorize-group-list = yes
### [domains] and [domain=domain_name] stanzas

The [domains] stanza contains a list of domains. Each domain specified under this stanza must have its own [domain=domain_name] stanza. For example:

```
[domains]
domain = d
domain = mydomain
```

The stanza entries for configuring multiple domains are located in the [domains] and the [domain=domain_name] stanzas of this configuration file:

- `ivmgrd.conf` for the policy server `pdmgrd`

### [domains] stanza

```
domain = domain_name
```

- **Name of the domain that was created.** The `domain_name` value is an alphanumeric, case-sensitive string. String values are expected to be characters that are part of the local code set.
- This stanza entry is required when the user creates at least one domain.
- There is no default value.
- Example: `domain = mydomain`

### [domain = domain_name] stanza

```
database-path = fully_qualified_path
```

- **File name and location where the database resides for the domain listed.** The name of the database is the same as the domain name. The file type can be anything, but the extension is usually `.db`.
- The `fully_qualified_path` value represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows, file names cannot have these characters: a backward slash (`\`), a colon (`:`), a question mark (`?`), or double quotation marks (`"`). For UNIX, path and file names are case sensitive.
- **Note:** Editing this entry in the configuration file is not recommended.
- This stanza entry is required when the user creates at least one domain.
- Default value for UNIX: `/var/PolicyDirector/db/domain_name.db`
- Default value for Windows: `pd_install_path\db\domain_name.db`
- Example for Windows: `D:\programs\ibm\am\db\dname1.db`
<table>
<thead>
<tr>
<th>allowed-registry-substrings = dn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distinguished name (DN) substring that restricts into which registry locations users can be created in or be imported from.</td>
</tr>
<tr>
<td>The DN of the user being created or imported must contain the substring value specified. The DN substring value restrictions are registry dependent. Most user registries allow an alphanumeric, case-insensitive string. String values are expected to be characters that are part of the local code set.</td>
</tr>
<tr>
<td>You can specify one or more relative DNs to use when creating users. By specifying one or more substrings, you can restrict creating and importing users and groups to the relative DNs that are identified by the substrings. For example, you can specify the DN substring <code>dc=mkt</code> to restrict users who are created or imported into a domain named Marketing:</td>
</tr>
<tr>
<td>As management domain administrator, you must:</td>
</tr>
<tr>
<td>1. Manually add the <code>dn</code> value for each domain created, except the Management (policy server) domain.</td>
</tr>
<tr>
<td>2. Notify the domain administrator, after this key value pair is added, to add this string to the DN option when creating and importing users or groups.</td>
</tr>
<tr>
<td>This stanza entry is optional.</td>
</tr>
<tr>
<td>There is no default value.</td>
</tr>
<tr>
<td>Example: <code>allowed-registry-substrings = &quot;dc=mkt&quot;</code></td>
</tr>
</tbody>
</table>
**[ivacld] stanza**

The stanza entries for configuring authorization server-related information are located in the `[ivacld] stanza` in this configuration file:

- `ivacld.conf` for the authorization server `pdacl`

<table>
<thead>
<tr>
<th>[ivacld] stanza</th>
</tr>
</thead>
<tbody>
<tr>
<td>`tcp-req-port = {0</td>
</tr>
</tbody>
</table>

Transmission Control Protocol (TCP) port on which the server is listening for requests.

Valid values include:

- 0: Disable the port number.
- `port`: Enable the port number.

For `port`, use any valid port number. A valid port number is any positive number that is allowed by TCP/IP and that is not currently being used by another application. It is recommended that you use the default port number value, or use a port number over 1000 that is not currently being used.

This stanza entry is required.

Default value: 7136

Example: `tcp-req-port = 7136`

<table>
<thead>
<tr>
<th>pid-file = <code>fully_qualified_path</code></th>
</tr>
</thead>
</table>

Location and name of the PID file.

The `fully_qualified_path` value represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows, file names cannot have these characters: a backward slash (`\`), a colon (`:`), a question mark (`?`), or double quotation marks (`"`). For UNIX, path and file names are case sensitive.

This stanza entry is required.

Default value for Windows: `C:\pd_install_path\log\ivacld.pid`

Default value for UNIX: `/var/PolicyDirector/log/ivacld.pid`

Example for Windows: `pid-file = C:\pd\log\ivacld.pid`
**log-file = fullyQualifiedPath**

Location and name of the log file. Messages are redirected from STDOUT and STDERR and sent to the server log file as defined in the authorization server routing file (pdacldRouting). The authorization server relies on the routing file to determine the log file names and path.

At startup of the authorization server (pdacld), a check is made to see if the routing file exists. If it exists, the routing file is used and this stanza entry is ignored; otherwise, this stanza entry is used.

The **fullyQualifiedPath** value represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows, file names cannot have these characters: a backward slash (\), a colon (:), a question mark (?), or double quotation marks ("). For UNIX, path and file names are case sensitive.

During installation of Tivoli Access Manager, if you enabled Tivoli Common Directory to specify one common directory location for all your Base component log files, the default installation directory will be different. For example:

```
log-file = TCD/HPD/logs/msg__pdacld_utf8.log
```

The 3-character identifier used in the example is HPD, which specifies that the log files are for the Tivoli Access Manager Base component.

This stanza entry is required.

Default installation location for Windows:
```
pd_install_path\log\msg__pdacld_utf8.log
```

Default installation location for UNIX:
```
/var/PolicyDirector/log/msg__pdacld_utf8.log
```

Example for Windows without Tivoli Common Directory: `log-file = C:\log\msg__pdacld_utf8.log`

Example for UNIX with Tivoli Common Directory:
```
/PolicyDirector/TAMBase/HPD/logs/msg__pdacld_utf8.log
```

**unix-user = userName**

UNIX user account for this server. The server will run as this UNIX user account.

This stanza entry is required.

Default value: ivmgr

Example: `unix-user = ivmgr`

**unix-group = groupName**

UNIX group account for this server. The server will run as this UNIX user account.

This stanza entry is required.

Default value: ivmgr

Example: `unix-group = ivmgr`
### permit-unauth-remote-caller= \{true|false\}

Indication of whether authorization API clients should be authorized by the authorization server before their requests are processed.

Valid values include:

- **true**: Authorization API clients should not be authorized.

  **Warning**: Setting this to true exposes the policy database in the domain for all clients to read, not just those that have been properly authorized with membership in the remote-acl-users group. Depending upon the nature of the policy within the domain security, system planners must consider the ability for any client to read system-defined policy to be a security problem.

- **false**: Authorization API clients should be authorized.

This stanza entry is optional.

Default value: false

Example: `permit-unauth-remote-caller= false`

### logcfg =

**category**:\{stdout|stderr|file|pipe|remote\}\[[param=value][param=value]]

Enables logging and auditing for the application. Category, destination, and other parameters are used to capture Tivoli Access Manager auditing and logging events.

Each server provides its own event logging setting in its corresponding configuration file.

Valid values include:

- **audit.azn**: Category that indicates auditing of the authorization component.
- **log-agent**: Indicates that the destination where log-agent is one of stdout, stderr, file, pipe, or remote. For example:
  
  `audit.azn:file`

- **path = path**: Specifies the name and location of the log file that is used for the log-agent.

- **flush_interval = flush_interval**: Specifies the frequency for flushing log file buffers.

- **log_id =**: Specifies the identifier for directing events from additional categories to the same log-agent.

Remove the pound signs (#) at the beginning of the configuration file lines to enable authentication or authorization auditing (or both) for the application.

This stanza entry is optional.

There is no default value.

Example for configuring authentication and authorization auditing:

```plaintext
logcfg = audit.azn:file path=/var/PolicyDirector/audit/ivacld.log,flush_interval=20,log_id=PDAclAudit
logcfg = audit.authn:file log_id=PDAclAudit
```
[ivmgrd] stanza

The stanza entries for configuring the policy server and policy database (also known as the master authorization database) are located in the [ivmgrd] stanza in this configuration file:

- ivmgrd.conf for the policy server pdmgrd

<table>
<thead>
<tr>
<th>[ivmgrd] stanza</th>
</tr>
</thead>
<tbody>
<tr>
<td>unix-user = user_name</td>
</tr>
<tr>
<td>UNIX user account for this server. The server will run as this UNIX user account.</td>
</tr>
<tr>
<td>This stanza entry is required.</td>
</tr>
<tr>
<td>Default value: ivmgr</td>
</tr>
<tr>
<td>Example: unix-user = ivmgr</td>
</tr>
</tbody>
</table>

| unix-group = group_name |
| UNIX group account for this server. The server will run as this UNIX user account. |
| This stanza entry is required. |
| Default value: ivmgr |
| Example: unix-group = ivmgr |

| database-path = fully_qualified_path |
| Location and name of the policy database, also known as the master authorization database. The file type can be anything but the extension is usually .db. |
| The fully_qualified_path value represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows, file names cannot have these characters: a backward slash (\), a colon (:), a question mark (?), or double quotation marks ("). For UNIX, path and file names are case sensitive. |
| **Note:** Editing this stanza entry in the configuration file is not recommended. |
| This stanza entry is required |
| Default value for UNIX: /var/PolicyDirector/db/master_authzn.db |
| Default value for Windows: pd_install_path\db\master_authzn.db |
| Example for Windows: database-path = C:\pd\db\master_authzn.db |
**tcp-req-port = \{0|port\}**

TCP port on which the server is listening for requests.

Valid values include:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disables the port number.</td>
</tr>
<tr>
<td>port</td>
<td>Enables the port number.</td>
</tr>
</tbody>
</table>

For `port`, use any valid port number. A valid port number is any positive number that is allowed by TCP/IP and that is not currently being used by another application. It is recommended that you use the default port number value, or use a port number over 1000 that is currently not being used.

This stanza entry is required.

Default value: 8135

Example: `tcp-req-port = 8135`

**auto-database-update-notify = \{yes|true|no|false\}**

Indication of automatic or manual update notification for authorization database replicas.

Value values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>Enable automatic update notification. This automatic setting is appropriate for environments where database changes are few and infrequent. When you configure update notification to be automatic, you must also correctly configure the <code>max-notifier-threads=</code> and <code>notifier-wait-time=</code> stanza entries.</td>
</tr>
<tr>
<td>false</td>
<td>Enable manual update notification.</td>
</tr>
</tbody>
</table>

This stanza entry is required.

Default value: yes

Example: `auto-database-update-notify = yes`

**max-notifier-threads = number_threads**

Maximum number of event notifier threads. The policy server is responsible for synchronizing all database replicas in the secure domain. When a change is made to the master database, notification threads do the work of announcing this change to all replicas. Each replica then has the responsibility to download the new information from the master.

When the update notification stanza entry is set to (yes), you must correctly configure this stanza entry and also the `notifier-wait-time=` stanza entry.

For `number_threads`, generally the value should be set to equal the number of existing replicas. Specify a valid integer positive whole number. Valid range for the number of threads is from 1 to 128 threads.

This stanza entry is required when `auto-database-update-notify = yes`.

Default value: 10

Example: `max-notifier-threads = 20`. 
<table>
<thead>
<tr>
<th>notifier-wait-time = <code>time_seconds</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (in seconds) the authorization policy database is idle before notification is sent to replicas. When the policy server is instructed to make a change to the master authorization database, it waits for a default period of time before sending out notifications to database replicas. This time delay is reset with each subsequent change to the database.</td>
</tr>
<tr>
<td>When the update notification stanza entry is set to (yes), you must correctly configure this stanza entry and also the <code>max-notifier-threads=</code> stanza entry.</td>
</tr>
<tr>
<td>Valid value include:</td>
</tr>
<tr>
<td><code>number_seconds</code></td>
</tr>
<tr>
<td>The number of seconds the authorization policy database is idle before notification is sent to replicas.</td>
</tr>
<tr>
<td>This stanza entry is required when <code>auto-database-update-notify</code> = yes.</td>
</tr>
<tr>
<td>Default value: 15 seconds</td>
</tr>
<tr>
<td>Example: notifier-wait-time = 30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>pid-file = <code>fully_qualified_path</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Location and name of the PID file.</td>
</tr>
<tr>
<td>The <code>fully_qualified_path</code> value represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows, file names cannot have these characters: a backward slash (), a colon (:), a question mark (?), or double quotation marks (&quot;). For UNIX, path and file names are case sensitive.</td>
</tr>
<tr>
<td>This stanza entry is required.</td>
</tr>
<tr>
<td>Default value for UNIX:</td>
</tr>
<tr>
<td><code>/var/PolicyDirector/log/ivmgrd.pid</code></td>
</tr>
<tr>
<td>Default value for Windows:</td>
</tr>
<tr>
<td><code>pd_install_path\log\ivmgrd.pid</code></td>
</tr>
<tr>
<td>Example for UNIX: pid-file = <code>/var/PolicyDirector/log/ivmgrd.pid</code></td>
</tr>
</tbody>
</table>
**log-file = fullyQualifiedPath**

Location and name of the log file. Messages are redirected from STDOUT and STDERR and sent to the server log file as defined in the policy server routing file (pdmgrd_routing). The policy server relies on the routing file to determine the log file names and path.

At startup of the policy server (pdmgrd), a check is made to see if the routing file exists. If it exists, the routing file is used and this stanza entry is ignored; otherwise, this stanza entry is used.

The `fullyQualifiedPath` value represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows, file names cannot have these characters: a backward slash (`\`), a colon (`:`), a question mark (`?`), or double quotation marks (`"`). For UNIX, path and file names are case sensitive.

This stanza entry is required.

Default installation location for Windows:
```
pd_install_path\log\msg__pdmgrd_utf8.log
```

Default installation location for UNIX:
```
/var/PolicyDirector/log/msg__pdmgrd_utf8.log
```

During installation of Tivoli Access Manager, if you enabled Tivoli Common Directory to specify one common directory location for all your log files, the default installation directory will be different. For example:
```
log-file = TCD_directory/HPD/logs
/msg__pdmgrd_utf8.log
```

The 3-character identifier used in the example is HPD, which specifies the log files are for the Tivoli Access Manager Base component:

Example for Windows without Tivoli Common Directory: log-file = `C:\pd\log\msg__pdmgrd_utf8.log`

**ca-cert-download-enabled = \{yes\|no\}**

Indication of whether other Tivoli Access Manager runtimes can download the root CA certificate automatically. Set this value by using the `mgrsslcfg` configuration utility.

Valid values include:
```
yes  Allow clients to download the root CA certificate automatically.
no   Does not allow clients to download the root CA certificate.
```

This stanza entry is optional.

Default value: no

Example: `ca-cert-download-enabled = yes`
### standby = \{0\}|\{number\}

Number of standby policy servers (**pdmgrd**)

**Note:** The number of standby servers is generated and set by the configuration utility. Do not edit this stanza entry.

Valid values include:

- **0**  Zero indicates that no policy servers are standby servers.
- **number**  The number of standby policy servers. Use a number that is an integer positive whole number. Currently, this number is only 1.

This stanza entry is required.

Default value: 0

Example: `standby = 1`

### logcfg =

| category: \{[stdout|stderr|file|pipe|remote]\}\{[[param]=[value]]\}[[param]=[value]]\} |

Enables logging and auditing for the application. Category, destination, and other parameters are used to capture Tivoli Access Manager auditing and logging events.

Each server provides its own event logging setting in its corresponding configuration file.

Valid values include:

- **audit.azn**  Category that indicates auditing of the authorization component.
- **log-agent**  Indicates that the destination where `log-agent` is one of `stdout`, `stderr`, `file`, `pipe`, or `remote`. For example:
  ```
  audit.azn:file
  ```
- **path = path**  Specifies the name and location of the log file that is used for the `log-agent`.
- **flush_interval = flush_interval**  Specifies the frequency for flushing log file buffers.
- **log_id = log_id**  Specifies the identifier for directing events from additional categories to the same `log-agent`.

Remove the pound signs (#) at the beginning of the configuration file lines to enable authentication or authorization auditing (or both) for the application.

This stanza entry is optional.

There is no default value.

Example for configuring only authentication and authorization auditing:

```bash
logcfg = audit.azn:file path=/var/PolicyDirector/audit/
pdmgrd.log,flush_interval=20,log_id=PDMgrAudit
logcfg = audit.authn:file log_id=PDMgrAudit
#logcfg = audit.mgmt:file log_id=PDMgrAudit
```
[ldap] stanza

This stanza defines configuration key value pairs that are required for the Tivoli Access Manager servers to communicate with the LDAP registry server.

The value for the user registry stanza entry (ldap-server-config) is determined by the pd.conf file. The pd.conf file is created when the Tivoli Access Manager runtime component is configured.

The configuration key value pairs used only for the LDAP registry server are located in the 1dap.conf configuration file in the [ldap] stanza. The LDAP server stanza entries are described separately in "[ldap] stanza for ldap.conf" on page 287.

The configuration key value pairs for the server configuration files are located in the [ldap] stanza of each of these configuration files:
- ivmgrd.conf for the policy server pdmgrd
- ivacld.conf for the authorization server pdacld
- pdmgrproxyd.conf for the policy proxy server pdmgrproxyd
- Your resource managers’ configuration file for configured LDAP entries

The aznAPI.conf configuration file is provided with Tivoli Access Manager as a sample file for creating your own resource manager configuration file. Developers of service plug-ins should provide the standard functions. Before implementing service plug-ins, read and thoroughly understand the concepts discussed in the IBM Tivoli Access Manager for e-business Authorization C API Developer Reference.

<table>
<thead>
<tr>
<th>[ldap] stanza</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldap-server-config = fully_qualified_path</td>
</tr>
<tr>
<td>Location of the 1dap.conf configuration file.</td>
</tr>
<tr>
<td>The <em>fully_qualified_path</em> value represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows, file names cannot have these characters: a backward slash (), a colon (:), a question mark (?), or double quotation marks (&quot;). For UNIX, path and file names are case sensitive.</td>
</tr>
<tr>
<td>This stanza entry is required for ivmgrd.conf.</td>
</tr>
<tr>
<td>Default installation location for UNIX: /opt/PolicyDirector/etc/ldap.conf</td>
</tr>
<tr>
<td>Default installation location for Windows: C:\Program Files\Tivoli\PolicyDirector\etc\1dap.conf</td>
</tr>
<tr>
<td>Example for UNIX: ldap-server-config = /opt/PolicyDirector/etc/ldap.conf</td>
</tr>
</tbody>
</table>
prefer-readwrite-server = {yes|true|no|false}

Indication of whether the client can question the Read/Write LDAP server before querying any replica Read-only servers that are configured in the domain.

The default value can be different. For example, the default value for ivmgrd.conf is yes while the default value for ivacbld.conf is no.

Valid values include:
- **yes|true** Enables the client to be able to question the Read/Write LDAP server.
- **no|false** Disables the client. Anything other than yes|true, including a blank value, is interpreted as no|false.

This stanza entry is optional.

There is no default value. The default value is server dependent.

Example: prefer-readwrite-server = no

bind-dn = LDAP_dn

LDAP user distinguished name (DN) that is used when binding (signing on) to the LDAP server. The LDAP_dn value is created, based on the server name that was specified with the `-n server_name` option and the local host of the machine.

Use the `svrsslcfg` utility to set the LDAP_dn value.

This stanza entry is required when the configured user registry is LDAP.

For information on how to use this key=value pair for performance tuning purposes, see the *IBM Tivoli Access Manager for e-business Performance Tuning Guide*.

There is no default value. The default values are server-dependent.

Example for the policy server `pdmgrd`: bind-dn = cn=ivmgrd/master,cn=SecurityDaemons,secAuthority=Default

ssl-enabled = {yes|true|no|false}

Indication of whether to enable SSL communication with the LDAP server. The value for each server can be different, depending on how the server was configured.

If you specify that the authorization API (aznAPI) communicate with the server by using SSL, you must enable the SSL using this stanza entry. Also, you must specify an SSL keyfile name to use, the SSL IP port, the keyfile DN only if there are multiple keys in the file, and a keyfile password.

Valid values include:
- **yes|true** Enables SSL communication.
- **no|false** Disables SSL communication. Anything other than yes|true, including a blank value, is interpreted as no|false, and SSL will be automatically configured.

This stanza entry is required to enable SSL communication.

There is no default value. The default values are server-dependent.

Example: ssl-enabled = yes
ssl-keyfile = ldap-ssl-key-filename

SSL key file name and location. Use the SSL key file to handle certificates that are used in LDAP communication. The file type can be anything but the extension is usually .kdb.

The certificate files in a directory need to be accessible to the server user (or all users). Make sure that server user (for example, ivmgr) or all users have permission to access the .kdb file and the folder that contains the .kdb file.

A valid file name is alphanumeric, case-insensitive string. String values are expected to be characters that are part of the local code set. For Windows, file names cannot have these characters: a backward slash (\), a colon (:), a question mark (?), or double quotation marks.

This stanza entry is required only when the LDAP server is configured to perform client authentication (ssl-enabled = yes).

There is no default value. The default values are server-dependent.

Default location for Windows:

\pd_install_dir\keytab\server_name.kdb

Default location for UNIX:

/opt/PolicyDirector/keytab/server_name.kdb

Example for UNIX for the policy server pdmgrd: ssl-keyfile = /ldap52kdb/a17jsun.kdb

ssl-keyfile-dn = ldap-ssl-keyfile-label

Key label of the client personal certificate within the SSL key file. This key label is used to identify the client certificate that is presented to the LDAP server.

This stanza entry is required only when the LDAP server is configured to perform client authentication.

If the default policy server pdmgrd key database is being used, the default client certificate value is: PDLADAP

Example: ssl-keyfile-dn = "PDLADAP"
auth-using-compare = {yes|true|no|false}

Choice of whether ldap_compare() will be used instead of the ldap_bind() call to verify the password and authenticate the user. For those LDAP servers that allow it, a compare operation might perform faster than a bind operation. The value for each server can be different, depending on how the server was configured.

This option changes the method used by these aznAPI calls:
- azn_util_client_authenticate()
- azn_util_password_authenticate()

Valid values include:
- yes|true
  A compare operation will be used to authenticate LDAP users instead of a bind operation.
- no|false
  A bind operation will be used to authenticate LDAP users instead of a compare operation. Anything other than yes|true, including a blank value, is interpreted as no|false.

For information on how to use this key=value pair for performance tuning purposes, see the IBM Tivoli Access Manager for e-business Performance Tuning Guide.

This stanza entry is optional.

There is no default value. The default values are server-dependent.

Example: auth-using-compare = yes

port = port

Non-SSL IP port number that is used for communicating with the LDAP server. For port, use the port number configured for the LDAP server.

This stanza entry is required for the pdmgrproxyd and pdacld servers. It is not required for the policy server pdmgrd. Default value: 389

Example: port = 389

default-policy-override-support = {yes|true|no|false}

Indication of whether user-level policy support is allowed.

Valid values include:
- yes|true
  User policy support is disabled and only the global (default) policy will be checked. This option allows the user policy to not be checked, even if it is specified.
- no|false
  User policy support is enabled. When a user policy is specified by the administrator, it overrides the global policy. If no value is specified, default-policy-override-support = no becomes the value.

For information on how to use this key=value pair for performance tuning purposes, see the IBM Tivoli Access Manager for e-business Performance Tuning Guide.

This stanza entry is optional.

Default value: no

Example: default-policy-override-support = yes
### user-and-group-in-same-suffix = \{yes\|true\|no\|false\}

Indication of whether the groups, in which a user is a member, are defined in the same LDAP suffix as the user definition.

When a user is authenticated, the groups in which the user is a member must be determined in order to build a credential. Normally, all LDAP suffixes are searched to locate the groups of which the user is a member.

Valid values include:

- **yes\|true**: The groups are assumed to be defined in same LDAP suffix as the user definition. Only that suffix will be searched for group membership. This behavior can improve the performance of group lookup because only a single suffix is searched for group membership. This option should only be specified if group definitions are restricted to the same suffix as the user definition.

- **no\|false**: The groups might be defined in any LDAP suffix. Anything other than yes\|true, including a blank value, is interpreted as no\|false.

For information on how to use this key=value pair for performance tuning purposes, see the *IBM Tivoli Access Manager for e-business Performance Tuning Guide*.

This stanza entry is optional.

Default value: no

Example: user-and-group-in-same-suffix = yes

### cache-enabled = \{yes\|true\|no\|false\}

Indication of whether LDAP client-side caching is used to improve performance for similar LDAP queries.

Valid values include:

- **yes\|true**: Enables LDAP client-side caching.

- **no\|false**: Disables LDAP client-side caching. This value is the default value. Anything other than yes\|true, including a blank value, is interpreted as no\|false.

For information on how to use this key = value pair for performance tuning purposes, see the *IBM Tivoli Access Manager for e-business Performance Tuning Guide*.

This stanza entry is optional.

Default value: no

Example: cache-enabled = no
<table>
<thead>
<tr>
<th>Configuration Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cache-user-size = user_entries</code></td>
<td>Number of entries in the LDAP user cache. This stanza entry will be ignored if the cache is not enabled. The <code>user_entries</code> value is specified as an integer positive whole number. This stanza entry is optional. Default value: 256 Example: <code>cache-user-size = 1000</code></td>
</tr>
<tr>
<td><code>cache-group-size = group_entries</code></td>
<td>Number of entries in the LDAP group cache. This stanza entry will be ignored if the cache is not enabled. The <code>group_entries</code> value is specified as an integer positive whole number. This stanza entry is optional. Default value: 64 Example: <code>cache-group-size = 100</code></td>
</tr>
<tr>
<td><code>cache-policy-size = policy_entries</code></td>
<td>Number of entries in the LDAP policy cache. This stanza entry will be ignored if the cache is not enabled. The <code>policy_entries</code> value is specified as an integer positive whole number. This stanza entry is optional. Default value: 20 Example: <code>cache-policy-size = 50</code></td>
</tr>
<tr>
<td><code>cache-user-expire-time = number_seconds</code></td>
<td>Amount of time (in seconds) until a user entry in the cache is considered stale and is discarded. This stanza entry will be ignored if the cache is not enabled. Valid values include: <code>number_seconds</code> - The amount of time specified in number of seconds. Use a number that is an integer positive whole number. This stanza entry is optional. Default value: 30 Example: <code>cache-user-expire-time = 120</code></td>
</tr>
</tbody>
</table>
**cache-group-expire-time = number_seconds**

Amount of time (in seconds) until a group entry in the cache is considered stale and is discarded. This stanza entry will be ignored if the cache is not enabled.

Valid values include:

- `number_seconds`

  The amount of time specified in number of seconds. Use a number that is an integer positive whole number.

This stanza entry is optional.

Default value: 300 (5 minutes)

Example: `cache-group-expire-time = 600`

**cache-policy-expire-time = number_seconds**

Amount of time (in seconds) until a policy entry in the cache is considered stale and is discarded. This stanza entry will be ignored if the cache is not enabled.

Valid values include:

- `number_seconds`

  The amount of time specified in number of seconds. Use a number that is an integer positive whole number.

This stanza entry is optional.

Default value: 30

Example: `cache-policy-expire-time = 60`

**cache-group-membership = {yes|true|no|false}**

Indication of whether group membership information is cached. This stanza entry is ignored if the cache is not enabled.

Valid values include:

- `yes|true`

  Group membership is cached.

- `no|false`

  Group membership is not cached. Anything other than `yes|true`, including a blank value, is interpreted as `no|false`.

This stanza entry is optional.

Default value: `yes|true`

Example: `cache-group-membership = no`
### cache-use-user-cache = [yes|true|no|false]

Indication of whether to use the user cache information. This stanza entry is ignored if the cache is not enabled.

Valid values include:
- **yes** | **true**
  - Use user information from the cache.
- **no** | **false**
  - Do not use user information from the cache. Anything other than **yes** | **true**, including a blank value, is interpreted as **no** | **false**.

This stanza entry is optional.

Default value: **yes** | **true**

Example: `cache-use-user-cache = no`

### max-search-size = [0|number_entries]

Limit for the maximum search size, specified as the number of entries, that can be returned from the LDAP server. The value for each server can be different, depending on how the server was configured.

Valid values include:
- **0**
  - The number is unlimited; there is no limit to the maximum search size.
- **number_entries**
  - The maximum number of entries for search, specified as an integer whole number. This value can be limited by the LDAP server itself.

This stanza entry is optional.

The default value is server-dependent but defaults to 2048 if not configured.

Example: `max-search-size = 2048`

### timeout = [0|number_seconds]

Amount of time (in seconds) that is allowed for authentication or search operations before the LDAP server is considered to be down. If specified, a value for the stanza entries `authn-timeout` or `search-timeout` overrides the value of this stanza entry.

**Note:** Do not specify this stanza entry in the `ldap.conf` server configuration file.

Valid values include:
- **0**
  - No timeout is allowed.
- **number_seconds**
  - The number of seconds allowed for authentication or search, specified as a positive integer whole number. There is no range limitation for timeout values.

This stanza entry is optional.

Default value: 0

Example: `timeout = 0`

### authn-timeout = [0|number_seconds]

Appendix A. Server configuration file reference 285
| Amount of time (in seconds) that will be allowed for authentication operations before the LDAP server is considered to be down. If specified, this value overrides any value of `timeout` for authentication operations. **Note:** Do not specify this stanza entry in the `ldap.conf` server configuration file.  
| Valid values include:  
| 0 | No timeout is allowed.  
| `number_seconds` | The specified number of seconds allowed for authentication operations, specified as an integer positive whole number. There is no range limitation for timeout values.  
| This stanza entry is optional.  
| Default value: 0  
| Example: `authn-timeout = 0` |

| `search-timeout = {0|number_seconds}` | Amount of time (in seconds) that will be allowed for search operations before the LDAP server is considered to be down. If specified, this value overrides any value of `timeout` for search operations. **Note:** Do not specify this stanza entry in the `ldap.conf` server configuration file.  
| Valid values include:  
| 0 | No timeout is allowed.  
| `number_seconds` | The specified number of seconds allowed for search operations, specified as an integer positive whole number. There is no range limitation for timeout values.  
| This stanza entry is optional.  
| Default value: 0  
| Example: `search-timeout = 0` |
[ldap] stanza for ldap.conf

This stanza defines configuration key value pairs that are required to communicate with the LDAP registry server. For example, you can find the configuration keys and values for LDAP failover, including the use of master and replica servers, in this stanza of the configuration file.

The user registry value is determined by the pd.conf file. The pd.conf file is created when the Tivoli Access Manager runtime component is configured.

For information on how to use the key=value pairs in this stanza for performance tuning purposes, see the IBM Tivoli Access Manager for e-business Performance Tuning Guide.

<table>
<thead>
<tr>
<th>[ldap] stanza</th>
</tr>
</thead>
<tbody>
<tr>
<td>enabled = (yes</td>
</tr>
<tr>
<td>Indication of whether LDAP is being used as the user registry. Only one user registry can be specified at a time.</td>
</tr>
<tr>
<td>If enabled, other required stanza entries are an LDAP server host name, and port with which to bind to the server, a bind user DN, and bind user password (obfuscated).</td>
</tr>
<tr>
<td>Valid values include:</td>
</tr>
<tr>
<td>yes</td>
</tr>
<tr>
<td>Enables LDAP user registry support.</td>
</tr>
<tr>
<td>no</td>
</tr>
<tr>
<td>Disables LDAP user registry support and indicates that LDAP is not the user registry being used. Anything other than yes</td>
</tr>
<tr>
<td>This stanza entry is required when LDAP is the user registry.</td>
</tr>
<tr>
<td>The default value can be different, depending on how the server is configured.</td>
</tr>
<tr>
<td>Example: enabled = yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>host = host_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host name of the LDAP server. Valid values for host_name include any valid Internet Protocol (IP) host name.</td>
</tr>
<tr>
<td>The host_name value is taken from the pd.conf file. The pd.conf file is created when the Tivoli Access Manager runtime component is configured on the machine.</td>
</tr>
<tr>
<td>Use the svrsslcfg utility to set the host_name value when the configured Policy Director user registry is LDAP.</td>
</tr>
<tr>
<td>This stanza entry is required.</td>
</tr>
<tr>
<td>There is no default value. The value is taken from the pd.conf file.</td>
</tr>
<tr>
<td>Examples of host names:</td>
</tr>
<tr>
<td>host = libra</td>
</tr>
<tr>
<td>host = libra.dallas.ibm.com</td>
</tr>
</tbody>
</table>
**cache-enabled** = \{yes|true|no|false\}

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indication of whether LDAP client-side caching is used to improve performance for similar LDAP queries.</td>
<td>Valid values include:</td>
</tr>
<tr>
<td>yes</td>
<td>Enables LDAP client-side caching.</td>
</tr>
<tr>
<td>true</td>
<td>Enables LDAP client-side caching.</td>
</tr>
<tr>
<td>no</td>
<td>Disables LDAP client-side caching. This value is the default value.</td>
</tr>
<tr>
<td>false</td>
<td>Disables LDAP client-side caching. Anything other than yes</td>
</tr>
</tbody>
</table>

For information on how to use this `key = value` pair for performance tuning purposes, see the *IBM Tivoli Access Manager for e-business Performance Tuning Guide*.

This stanza entry is optional.

Default value: no

Example: `cache-enabled = no`

**port** = `port`

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-SSL IP port number that is used for communicating with the LDAP server. For <strong>port</strong>, use the value configured for the LDAP server.</td>
<td>This stanza entry is required.</td>
</tr>
</tbody>
</table>

Default value: 389

Example: `port = 389`

**ssl-port** = `port`

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL IP port that is used to connect to the LDAP server. For <strong>port</strong>, use any valid port number. A valid port number is any positive number that is allowed by TCP/IP and that is not currently being used by another application.</td>
<td>This stanza entry is required only when the LDAP server is configured to perform client authentication (ssl-enabled = yes).</td>
</tr>
</tbody>
</table>

Default value: 636

Example: `ssl-port = 636`
**LdapSSL** = \{ssl|noss\}

Indication of whether to enable SSL communication with the LDAP server. The value for each server can be different, depending on how the server was configured.

Valid values include:
- **ssl**: Enables SSL communication. SSL will be automatically configured.
- **noss**: Disables SSL communication. Anything other than **ssl**, including a blank value, is interpreted as **noss**.

This stanza entry is optional.

The default value is server dependent.

Example: LdapSSL = noss

**LdapSSLKeyFile** = *ldap-ssl-key-filename*

SSL key file name and location. Use the SSL key file to handle certificates that are used in LDAP communication. The file type can be anything but the extension is usually `.kdb`.

The certificate files in a directory need to be accessible to the server user (or all users). Make sure that server user (for example, `ivmgr`) or all users have permission to access the `.kdb` file and the folder that contains the `.kdb` file.

The file name and location value represents an alphanumeric, case-insensitive string. String values are expected to be characters that are part of the local code set. The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows, file names cannot have these characters: a backward slash (`\`), a colon (`:`), a question mark (`?`), or double quotation marks (`"`).

This stanza entry is required only when LdapSSL = yes.

Default value is server dependent.

Default value for Windows:

```
pd_install_dir\keytab\ivmgrd.kdb
```

Default value for UNIX:

```
/opt/PolicyDirector/keytab/ivmgrd.kdb
```

Example for UNIX: LdapSSLKeyFile = `/opt/PolicyDirector/keytab/ivmgrd.kdb`

**LdapSSLKeyFileDn** = *keyLabel*

Key label of the client personal certificate within the SSL key file. This key label is used to identify the client certificate that is presented to the LDAP server. This stanza entry is used when the LDAP server is configured to perform client authentication.

This stanza entry is required only when LdapSSL = yes.

There is no default value.

Example: LdapSSLKeyFileDn = "PD_LDAP"
**LdapSSLKeyFilePwd = ldap-ssl-keyfile-password**

Password to access the SSL key file.

**Note:** The password associated with the default SSL keyfile is *key4ssl*.

This stanza entry is required only when *LdapSSL = yes*.

There is no default value.

Example: *LdapSSLKeyFilePwd = mysslpwd*

**replica = ldap-server, port, type, pref**

Definition of the LDAP user registry replicas in the domain where:

- *ldap-server* is the network name of the server.
- *port* is the port number for the LDAP server. A valid port number is any positive number that is allowed by TCP/IP and that is not currently being used by another application.
- *type* is one of readonly or readwrite.
- *pref* is a number from 1 to 10 (10 is the highest preference).

This stanza entry is optional.

Default value is that no replicas are specified.

Example of one replica specified and two replicas commented out:

*replica = freddy,390,readonly,1*

#*replica = barney,391,readwrite,2*

#*replica = benny,392,readwrite,3*

**dynamic-groups-enabled = {yes|true|no|false}**

Indication of whether dynamic groups are supported.

**Note:** This stanza entry can only be used in the *ldap.conf* configuration file.

Valid values include:

- **yes|true**
  Tivoli Access Manager attempts to resolve dynamic group membership.

- **no|false**
  Tivoli Access Manager does not attempt to resolve dynamic group membership. Anything other than *yes|true*, including a blank value, is interpreted as *no|false*.

This stanza entry is optional.

Default value: no

Example: *dynamic-groups-enabled = no*
<table>
<thead>
<tr>
<th><strong>ignore-suffix = suffix_dn</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP server suffix that is to be ignored when searching for user and group information.</td>
</tr>
<tr>
<td><strong>Note:</strong> This stanza entry can only be used in the ldap.conf configuration file.</td>
</tr>
<tr>
<td>The <em>suffix_dn</em> value specifies the suffix distinguished name (DN) that you want to be ignored. Repeat this stanza entry for each suffix you want to be ignored. For example, if you specify <code>ignore-suffix = o=tivoli,c=us</code>, any user or group that includes <code>o=tivoli,c=us</code> as part of the DN will be ignored.</td>
</tr>
<tr>
<td>This stanza entry is optional.</td>
</tr>
<tr>
<td>Default value: All defined suffixes are searched.</td>
</tr>
<tr>
<td>Example: <code>ignore-suffix = o=tivoli,c=us</code></td>
</tr>
</tbody>
</table>
[manager] stanza

The stanza entries for configuring the master server settings are located in the [manager] stanza of each of these configuration files:
- `ivacld.conf` for the authorization server `pdacld`
- `pd.conf` when you use the authorization server `pdacld`
- `pdmgrproxyd.conf` for the policy proxy server `pdmgrproxyd`

<table>
<thead>
<tr>
<th>[manager] stanza</th>
<th>master-host = server_hostname</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Host name of the Tivoli Access Manager server. Examples of valid host names:</td>
</tr>
<tr>
<td></td>
<td>• mycomputer.city.company.com</td>
</tr>
<tr>
<td></td>
<td>• mycomputer</td>
</tr>
<tr>
<td></td>
<td>This stanza entry is required.</td>
</tr>
<tr>
<td></td>
<td>There is no default value.</td>
</tr>
<tr>
<td></td>
<td>Example: master-host = ammaster</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[manager] stanza</th>
<th>master-port = port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TCP port on which the server is listening for requests. This value is created and set by one of these utilities:</td>
</tr>
<tr>
<td></td>
<td>• For <code>pd.conf</code>, the value is set by using the <code>bassslcfg</code> utility.</td>
</tr>
<tr>
<td></td>
<td>• For all other configuration files, the value is set by using the <code>svrsslcfg</code> utility.</td>
</tr>
<tr>
<td></td>
<td>For <code>port</code>, use any valid port number. A valid port number is any positive number that is allowed by TCP/IP and that is not currently being used by another application. It is recommended that you use the default port number value, or use a port number over 1000 that is currently not being used.</td>
</tr>
<tr>
<td></td>
<td>This stanza entry is required.</td>
</tr>
<tr>
<td></td>
<td>The default value is server-dependent.</td>
</tr>
<tr>
<td></td>
<td>Example: master-port = 7135</td>
</tr>
</tbody>
</table>
management-domain = [default|domain_name]

Name of the management domain. This value is created and set by one of these utilities:

- For pd.conf, the value is set by the bassslcfg utility. For the pd.conf configuration file: if this value is not in the configuration file, operations that rely on its presence will fail.
- For other configuration files, the value is set by the svrsslcfg utility.

Valid values include:

**Default**

Specifies the Management domain. This value is the default value for all servers.

**domain_name**

Specifies the user-specified domain. Use this value when you configure your own name for the domain.

The *domain_name* value is an alphanumeric, case-sensitive string. String values are expected to be characters that are part of the local code set.

Valid characters for domain names for U.S. English are the letters a-Z, the numbers 0-9, a period (.), an underscore (_), a plus sign (+), a hyphen (-), an at sign (@), an ampersand (&), and an asterisk (*). You cannot use a space in the domain name.

This stanza entry is required.

Default value for all servers: Default

Example: management-domain = mymgmtdomain
[meta-info] stanza

The stanza entry for configuring Tivoli Access Manager version information is located in the [meta-info] stanza of each of these configuration files:

- ivmgrd.conf for the policy server pdmgrd
- ivacld.conf for the authorization server pdacld
- pd.conf when you use the authorization server pdacld
- pdmgrproxyd.conf for the policy proxy server pdmgrproxyd

<table>
<thead>
<tr>
<th>[meta-info] stanza</th>
</tr>
</thead>
<tbody>
<tr>
<td>version = number</td>
</tr>
</tbody>
</table>

Version of the Tivoli Access Manager in decimal format. For example, Version 5.1 of Tivoli Access Manager (Base) is represented as 1296 when converted to decimal.

Note: This value is generated; do not change it.

The number value is automatically filled in by the application runtime and must not be edited. Tivoli Access Manager runtime determines what the value should be in the file. In some cases, Tivoli Access Manager uses the version key to determine if the appropriate configuration file has been upgraded to the level of the Tivoli Access Manager Base component.

This stanza entry is required.

Default value for v5.1: 1296

Example: version = 1296
**[pdmgrproxyd] stanza**

The stanza entries for configuring the policy proxy server are located in:
- `pdmgrproxyd.conf` for the policy proxy server `pdmgrproxyd`

```
unix-user = ivmgr
unix-group = ivmgr
```

### [pdmgrproxyd] stanza

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tcp-req-port</code></td>
<td>TCP port on which the server is listening for requests.</td>
</tr>
<tr>
<td><code>0</code></td>
<td>Disables the port number.</td>
</tr>
<tr>
<td><code>port</code></td>
<td>Enables the port number. For <code>port</code>, use any valid port number. A valid</td>
</tr>
<tr>
<td></td>
<td>port number is any positive number that is allowed by TCP/IP and</td>
</tr>
<tr>
<td></td>
<td>that is not currently being used by another application. It is</td>
</tr>
<tr>
<td></td>
<td>recommended that you use the default port number value, or use a</td>
</tr>
<tr>
<td></td>
<td>port number over 1000 that is currently not being used.</td>
</tr>
</tbody>
</table>

This stanza entry is required.

Default value: 8138

Example: `tcp-req-port = 8138`

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pid-file</code></td>
<td>Location and name of the PID file.</td>
</tr>
<tr>
<td></td>
<td>The <code>fully_qualified_path</code> value represents an alphanumeric string. String</td>
</tr>
<tr>
<td></td>
<td>values are expected to be characters that are part of the local code set.</td>
</tr>
<tr>
<td></td>
<td>The set of characters permitted in a file name can be determined by the</td>
</tr>
<tr>
<td></td>
<td>file system and by the local code set. For Windows, file names cannot</td>
</tr>
<tr>
<td></td>
<td>have these characters: a backward slash (<code>\</code>), a colon (<code>:</code>), a question</td>
</tr>
<tr>
<td></td>
<td>mark (<code>?</code>), or double quotation marks (<code>&quot;</code>). For UNIX, path and file names</td>
</tr>
<tr>
<td></td>
<td>are case sensitive.</td>
</tr>
</tbody>
</table>

This stanza entry is required.

Default value for Windows: `pd_install_path\log\pdmgrproxyd.pid`

Default value for UNIX: `/var/PolicyDirector/log/ pdmgrproxyd.pid`

Example for Windows: `pid-file = C:\pd\log\pdmgrproxyd.pid`
<table>
<thead>
<tr>
<th><strong>log-file</strong> = <em>fullyQualifiedPath</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>Location and name of the log file. Messages are redirected from STDOUT and STDERR and sent to the server log file as defined in the policy proxy server routing file (pdmgrproxyd_routing). The policy proxy server relies on the routing file to determine the log file names and path.</td>
</tr>
<tr>
<td>At startup of the policy proxy server (<em>pdmgrproxyd</em>), a check is made to see if the routing file exists. If it exists, the routing file is used and this stanza entry is ignored; otherwise, this stanza entry is used.</td>
</tr>
<tr>
<td>The <em>fullyQualifiedPath</em> value represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows, file names cannot have these characters: a backward slash (), a colon (:), a question mark (?), or double quotation marks (&quot;). For UNIX, path and file names are case sensitive.</td>
</tr>
<tr>
<td>This stanza entry is required.</td>
</tr>
<tr>
<td>Default installation location for Windows:</td>
</tr>
<tr>
<td><em>pd_install_path</em>\log\msg__pdmgrproxyd_utf8.log</td>
</tr>
<tr>
<td>Default installation location for UNIX:</td>
</tr>
<tr>
<td>/var/PolicyDirector/log/msg__pdmgrproxyd_utf8.log</td>
</tr>
<tr>
<td>During installation of Tivoli Access Manager, if you enabled Tivoli Common Directory to specify one common directory location for all your log files, the default installation directory will be different. For example:</td>
</tr>
<tr>
<td><em>log-file</em> = <em>TCD_directory</em>/HPD/logs</td>
</tr>
<tr>
<td>/msg__pdmgrproxyd_utf8.log</td>
</tr>
<tr>
<td>The 3-character identifier used in the example is HPD, which specifies the log files are for the Tivoli Access Manager Base component.</td>
</tr>
<tr>
<td>Example for Windows without Tivoli Common Directory: <em>log-file</em> =</td>
</tr>
<tr>
<td>C:\pd\log\msg__pdmgrproxyd_utf8.log</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>unix-user</strong> = <em>user_name</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX user account for this server. UNIX group and user names are different items and both can have the same value. The user name is set as the user owner of the proxy manager files. The validity of the user name specified depends on the requirements of the UNIX platform.</td>
</tr>
<tr>
<td>This stanza entry is required when working with UNIX user accounts.</td>
</tr>
<tr>
<td>Default value: ivmgr</td>
</tr>
<tr>
<td>Example: unix-user = ivmgr</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>unix-group</strong> = <em>group_name</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX group account for this server. UNIX group and user names are different items and both can have the same value. The user name is set as the group owner of the policy proxy server (<em>pdmgrproxyd</em>) files. The validity of the group name specified depends on the requirements of the UNIX platform.</td>
</tr>
<tr>
<td>This stanza entry is required when working with UNIX user accounts.</td>
</tr>
<tr>
<td>Default value: ivmgr</td>
</tr>
<tr>
<td>Example: unix-group = ivmgr</td>
</tr>
<tr>
<td>cache-database = {yes</td>
</tr>
<tr>
<td>--------------------------</td>
</tr>
<tr>
<td>Indication of whether in-memory caching of the policy database is enabled.</td>
</tr>
<tr>
<td>Valid values include:</td>
</tr>
<tr>
<td>yes</td>
</tr>
<tr>
<td>no</td>
</tr>
<tr>
<td>This stanza entry is required.</td>
</tr>
<tr>
<td>Default value: no</td>
</tr>
<tr>
<td>Example: cache-database = yes</td>
</tr>
</tbody>
</table>
[pdrte] stanza

For the policy server, when the PDMgr package is installed, the policy server automatically starts after each system reboot. For the authorization server, when the PDAcld package is installed, the authorization server daemon automatically starts after each system reboot.

The stanza entries for automating server startup when using any of the user registries are located in the [pdrte] stanza of this configuration file:
- pd.conf when you use the authorization server pdacld

When you use the authorization server, pdacld for Tivoli Access Manager, you must have the pd.conf configuration file.

### [pdrte] stanza

| configured | {yes|no} |
|------------|--------|
| Indication of whether the Tivoli Access Manager runtime package has been configured. |
| **Note:** This value is generated; do not change it. |
| Valid values include: |
| yes | The runtime package has been configured. |
| no | The runtime package has not been configured. |
| This stanza entry is required. |
| There is no default value. |
| Example: configured = no |

| user-reg-type | {ldap|domino|active_directory} |
|---------------|--------------------------------|
| User registry type. |
| **Note:** This value is generated during configuration; do not change it. |
| Valid values for Windows: |
| ldap | LDAP is configured as the user registry. |
| domino | Lotus Domino is configured as the user registry. |
| active_directory | Microsoft Active Directory is configured as the user registry. |
| Valid values for UNIX: |
| ldap | LDAP is configured as the user registry. |
| active_directory | Microsoft Active Directory is configured as the user registry. |
| This stanza entry is required. |
| There is no default value. |
| Example: user-reg-type = ldap |

<table>
<thead>
<tr>
<th>user-reg-server</th>
<th>server_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>User registry server name.</td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong> This value is generated during configuration; do not change it.</td>
<td></td>
</tr>
<tr>
<td>A valid server_name is any alphanumeric non-case sensitive name that is allowed by the TCP/IP protocol.</td>
<td></td>
</tr>
<tr>
<td>This stanza entry is required.</td>
<td></td>
</tr>
<tr>
<td>There is no default value.</td>
<td></td>
</tr>
<tr>
<td>Example: user-reg-server = libra</td>
<td></td>
</tr>
<tr>
<td>User Registry Host Name</td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>user-reg-host = hostname</strong></td>
<td></td>
</tr>
<tr>
<td>User registry host name.</td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong> This value is generated during configuration; do not change it.</td>
<td></td>
</tr>
<tr>
<td>Examples of valid host names:</td>
<td></td>
</tr>
<tr>
<td>* mycomputer.city.company.com</td>
<td></td>
</tr>
<tr>
<td>* mycomputer</td>
<td></td>
</tr>
<tr>
<td>This stanza entry is required.</td>
<td></td>
</tr>
<tr>
<td>There is no default value.</td>
<td></td>
</tr>
<tr>
<td>Example: user-reg-host = libra</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Registry Host Port</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>user-reg-hostport = port</strong></td>
</tr>
<tr>
<td>Non-SSL IP port number that is used for communicating with the user registry server.</td>
</tr>
<tr>
<td><strong>Note:</strong> This value is generated during configuration; do not change it.</td>
</tr>
<tr>
<td>The <strong>port</strong> is any valid port number, meaning any positive number that is allowed by TCP/IP and that is not currently being used by another application.</td>
</tr>
<tr>
<td>This stanza entry is required.</td>
</tr>
<tr>
<td>There is no default value.</td>
</tr>
<tr>
<td>Example: user-reg-hostport = 389</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Boot Start IVmgrd</th>
</tr>
</thead>
<tbody>
<tr>
<td>**boot-start-ivmgrd = {yes</td>
</tr>
<tr>
<td>Indication of whether to start the policy server (<strong>pdmgrd</strong>) at system boot.</td>
</tr>
<tr>
<td>Valid values include:</td>
</tr>
<tr>
<td>* yes Start the policy server <strong>pdmgrd</strong> at system boot.</td>
</tr>
<tr>
<td>* no Do not start the policy server <strong>pdmgrd</strong> at system boot.</td>
</tr>
<tr>
<td>This stanza entry is required only for UNIX systems.</td>
</tr>
<tr>
<td>Default value: no</td>
</tr>
<tr>
<td>Example: boot-start-ivmgrd = yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Boot Start IVacld</th>
</tr>
</thead>
<tbody>
<tr>
<td>**boot-start-ivacld = {yes</td>
</tr>
<tr>
<td>Indication of whether to start the authorization server (<strong>pdacld</strong>) at system boot.</td>
</tr>
<tr>
<td>Valid values include:</td>
</tr>
<tr>
<td>* yes Start the <strong>pdacld</strong> authorization server at system boot.</td>
</tr>
<tr>
<td>* no Do not start the <strong>pdacld</strong> authorization server at system boot.</td>
</tr>
<tr>
<td>This stanza entry is required only for UNIX systems.</td>
</tr>
<tr>
<td>Default value: no</td>
</tr>
<tr>
<td>Example: boot-start-ivacld = yes</td>
</tr>
</tbody>
</table>
boot-start-pdproxyd = {yes|no}

Indication of whether to start the policy proxy server (pdmgrproxyd) at system boot.

Valid values include:

yes Start the policy proxy server pdmgrproxyd at system boot.
no Do not start the policy proxy server pdmgrproxyd at system boot.

This stanza entry is required only for UNIX systems.

Default value: no

Example: boot-start-pdproxyd = yes

tivoli_common_dir = fullyQualifiedNamePath

File name and location for message and trace log files. Indicates whether Tivoli Common Director is used.

The fullyQualifiedNamePath value represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows, file names cannot have these characters: a backward slash (\), a colon (:), a question mark (?), or double quotation marks ("). For UNIX, path and file names are case sensitive.

This stanza entry is required only if you configure the Tivoli Access Manager Java runtime environment for Tivoli Common Directory (TCD) logging.

Refer to "Common log file directories" on page 174 for more information about Tivoli Common Director.
**[pdwpm] stanza**

The stanza entry for configuring Tivoli Access Manager Web Portal Manager information is located in the `[meta-info] stanza of the pdwpm.conf configuration file.

This configuration file also lists the possible directories that can be used for the customized .gif files. A stanza entry is not used for listing the image file names and location.

<table>
<thead>
<tr>
<th><strong>[pdwpm] stanza</strong></th>
</tr>
</thead>
</table>

| authMethod = \{FORM|BASIC|SSO|TAI\} |
|-----------------------------|

Authentication method.

Valid values include:

- **FORM**: Use when FORM-based login is needed.
- **BASIC**: Use when basic authentication is needed.
- **SSO**: Use for single signon, when Web Portal Manager is junctioned behind WebSEAL.
- **TAI**: Use when Web Portal Manager is junctioned behind WebSEAL and WebSphere Application Server security is in use.

This stanza entry is required.

Default value: FORM

Example: authMethod = BASIC

| changePassword = \{true|false\} |
|--------------------------------|

Indication of whether to allow the Web Portal Manager user to change their password.

Valid values include:

- **true**: Display pages that allow Web Portal Manager users to change their password.
- **false**: Do not display pages that allow Web Portal Manager users to change their password.

Note that Web Portal Manager passwords must adhere to the password policies set by the administrator. By default, passwords must contain a minimum of eight characters (consisting of at least one number and four letters) and a maximum of two repeated characters.

This stanza entry is required.

Default value: true

Example: changePassword = false
**jrteProps = ** _fully_qualified_path_

File name and location of Tivoli Access Manager runtime environment properties file. This stanza entry requires the `jrteHost` stanza entry.

**Note:** This value is generated during configuration; do not change it.

The `fully_qualified_path` value is automatically filled in by the application runtime and must not be edited. Tivoli Access Manager runtime determines what the value should be in the file. In some cases, Tivoli Access Manager uses the version key to determine if the appropriate configuration file has been upgraded to the level of the Tivoli Access Manager Base.

This stanza entry is required.

Default installation location for Windows:
`pd_install_dir\java\export\pdwpm\pdwpm.properties`

Default installation location for UNIX:
`/opt/PolicyDirector/java/export/pdwpm/pdwpm.properties`

Example for Windows: `jrteProps = C:\Program Files\Tivoli\PolicyDirector\java\export\pdwpm\pdwpm.properties`

**jrteHost = ** _hostname_

Host name configured during Java Runtime installation.

**Note:** This value is generated during configuration; do not change it.

Examples of valid host names:
* mycomputer.city.company.com
* mycomputer

This stanza entry requires the `jrteProps` stanza entry.

The `hostname` value is automatically filled in during runtime configuration. The host name value is an alphanumeric, case-insensitive string. String values are expected to be characters that are part of the local code set. The dot (.) cannot be the last character of the host name.

This stanza entry is required.

There is no default value.

Example: `jrteHost = localhost`

**debug = ** `{true|false}`

Determines whether the trace is permitted to be displayed to standard out (stdout).

Valid values include:
- **true** Allows the trace information to be displayed to stdout.
- **false** Does not allow the trace information to be displayed to stdout.

This stanza entry is required.

Default value: **false**

Example: `debug = true`
[ssl] stanza

The [ssl] stanza in the configuration file defines the Secure Sockets Layer (SSL) configuration settings for the Tivoli Access Manager servers. The stanza entries for configuring Tivoli Access Manager SSL settings are located in the [ssl-info] stanza of each of these configuration files:

- ivmgrd.conf for the policy server pdmgrd
- ivacld.conf for the authorization server pdacld
- pd.conf when you use the authorization server pdacld
- pdmgrproxyd.conf for the policy proxy server pdmgrproxyd
- Your resource managers’ configuration file for configured SSL entries

The aznAPI.conf configuration file is provided with Tivoli Access Manager as a sample file for creating your own resource manager configuration file. Developers of service plug-ins should provide the standard functions. Before implementing service plug-ins, read and thoroughly understand the concepts discussed in the IBM Tivoli Access Manager for e-business Authorization C API Developer Reference.

<table>
<thead>
<tr>
<th>[ssl] stanza</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ssl-keyfile</strong> = <strong>ssl-key-path</strong></td>
</tr>
<tr>
<td>Path location and file name of the local system of the SSL key file. If the key value pair does not exist in the configuration file, the application will fail. The file extension can be anything but it is usually .kdb.</td>
</tr>
<tr>
<td>The certificate files in a directory need to be accessible to the server user (or all users). Make sure that server user (for example, ivmgr) or all users have permission to access the .kdb file and the folder that contains the .kdb file.</td>
</tr>
<tr>
<td>The file name and location value represents an alphanumeric, case-insensitive string. <strong>Note:</strong> The file name, including extension, is generated and set by the configuration utility. Do not edit this stanza entry.</td>
</tr>
<tr>
<td>This file is created and the value is set by using these utilities:</td>
</tr>
<tr>
<td>• For ivmgrd.conf, it is set by mgrsslcfg. The name and path are fixed for ivmgrd.conf.</td>
</tr>
<tr>
<td>• For pd.conf, it is set by basssslcfg. The name and path are fixed for pd.conf.</td>
</tr>
<tr>
<td>• For all others, it is set by svrsslcfg.</td>
</tr>
<tr>
<td>This stanza entry is required.</td>
</tr>
<tr>
<td>There is no default value. The name is set by the configuration program.</td>
</tr>
<tr>
<td>Default installation location for UNIX:</td>
</tr>
<tr>
<td>/var/PolicyDirector/keytab/server_name.kdb</td>
</tr>
<tr>
<td>Default installation location for Windows:</td>
</tr>
<tr>
<td>pd_install_path\keytab\server_name.kdb</td>
</tr>
<tr>
<td>Example for UNIX for the policy server pdmgrd: ssl-keyfile = /var/PolicyDirector/keytab/ivmgrd.kdb</td>
</tr>
</tbody>
</table>

Appendix A. Server configuration file reference 303
ssl-keyfile-stash = ssl-stash-path

Path location and file name of the SSL password stash file. The file extension can be anything, but it is usually .sth. The password is used to protect private keys in the key file. The password might be stored encrypted in the stash file.

The file name and location value represents an alphanumeric, case-insensitive string.

**Note:** The file name, including extension, is generated and set by the configuration utility. Do not edit this stanza entry.

This file is created and the value is set by using these utilities:

- For ivmgrd.conf, it is set by `mgrsslcfg`. The name and path are fixed for ivmgrd.conf.
- For pd.conf, it is set by `basslcfg`. The name and path are fixed for pd.conf.
- For all others, it is set by `svrsslcfg`. The path is defined by the `-d` option to the `svrsslcfg` utility. The name is defined by the `-n` option to the `svrsslcfg` utility.

This stanza entry is required.

There is no default value. The name is set by the configuration program.

Default installation location for UNIX:
`var/PolicyDirector/keytab/server_name.sth`

Default installation location for Windows:
`C:\pd_install_path\keytab\server_name.sth`

Example for Windows for the `pdmgrproxyd` server: `ssl-keyfile-stash = /var/PolicyDirector/keytab/pdmgrproxyd.sth`

ssl-keyfile-label = label

Label of key to use other than the default. Quotation marks surrounding the `label` value are not permitted.

There is no support for a user specified name. This label is created and the value is set by using one of these utilities:

- For ivmgrd.conf, it is set by `mgrsslcfg`.
- For pd.conf, this parameter does not apply.
- For all others, it is set by `svrsslcfg`.

This stanza entry is required.

There is no default value. The name is set by the configuration program.

Example for the policy server `pdmgrd`: `ssl-keyfile-label = PD Management Server`
### ssl-v3-timeout = number_seconds

Session timeout (in seconds) for SSL v3 connections between clients and servers. This timeout value controls how often a full SSL handshake is completed between Tivoli Access Manager clients and servers.

Valid values include:

<table>
<thead>
<tr>
<th>number_seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid range of values for the number of seconds is from 10-86400 seconds, where 86400 seconds is equal to 1 day. If you specify a number outside this range, the default number will be used.</td>
</tr>
</tbody>
</table>

This timeout value is created and the value is set by using these utilities:

- For ivmgrd.conf, it is set by `mgrsslcfg`.
- For pd.conf, it is set by `basslcfg`. The name and path is fixed for pd.conf.
- For all others, it is set by `svrsslcfg`. The path is defined by the `-d` option of the `svrsslcfg` utility. The name is defined by the `-n` option of `svrsslcfg`.

**Note:** Tivoli Access Manager components might not function with small timeout values in some network environments.

This stanza entry is required.

The default value is server-dependent.

Example: `ssl-v3-timeout = 7200`.

### ssl-io-inactivity-timeout = (0|number_seconds)

Duration (in seconds) that an SSL connection waits for a response before timing out. There is no one default value because the default value is set by the configuration program of each server.

Valid values include:

<table>
<thead>
<tr>
<th>number_seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 No timeout is allowed.</td>
</tr>
<tr>
<td>The timeout specified in number of seconds. There is no range limitation for timeout values.</td>
</tr>
</tbody>
</table>

The number of seconds value is set by using one of these utilities:

- For ivmgrd.conf, it is set by `mgrsslcfg`.
- For pd.conf, it is set by `basslcfg`.
- For all others, it is set by `svrsslcfg`.

This stanza entry is required.

The default value is server-dependent.

Example: `ssl-io-inactivity-timeout = 90`.

### ssl-maximum-worker-threads = number_threads

Number of threads that can be created by the server to handle incoming requests.

Valid values include:

<table>
<thead>
<tr>
<th>number_threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of threads that can be specified. The valid range must be equal to 1 or greater than 1. The maximum number varies because it is dependent on available system resources.</td>
</tr>
</tbody>
</table>

This stanza entry is required.

The default value is server-dependent.

Example: `ssl-maximum-worker-threads = 50`.
**ssl-pwd-life = number_days**

Password lifetime for the key database file, specified in the number of days. For automatic password renewal, the value for the lifetime of a password is controlled by the *number_days* value when the server is started. Valid values for the *number_days* is from 1 to 7,299 days.

For manual password renewal, the value is dictated by the value supplied to the `svrsslcfg -chgpwd` command. This value is also written into the appropriate configuration file.

**Note:** If a certificate and the password to the keyring database file containing that certificate are both expired, then the password must be refreshed first.

The number of days value is created and the value is set by using one of these utilities:

- For `ivmgrd.conf`, it is set by `mgrsslcfg`.
- For `pd.conf`, it is set by `basslcfg`.
- For all others, it is set by `svrsslcfg`.

This stanza entry is required.

The default value is server-dependent.

Example: `ssl-pwd-life = 183`

---

**ssl-cert-life = number_days**

Value for the lifetime (in number of days) of a certificate. Any issued or renewed certificates must use this value.

**Note:** Only the policy server `pdmgrd` uses this value.

For `ivmgrd.conf`, you set the *number_days* value by using the `mgrsslcfg` utility. The name and path are fixed for `ivmgrd.conf`. Use this utility to modify this value after initial configuration.

For *number_days*, use a positive whole number range between 1 and 7299.

To increase or decrease the value, change the value and restart the policy server `pdmgrd`. The new value is in effect only for certificates issued or renewed from that point on. If both the certificate and the password to the keyring database file that contains the certificate expire, the password must be refreshed first.

This stanza entry is required.

The default value is server-dependent.

Example: `ssl-cert-life = 365`
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ssl-auto-refresh</code></td>
<td>Indication of whether automatic refresh of the SSL certificate and the key</td>
</tr>
<tr>
<td></td>
<td>database file password occur. Valid values include:</td>
</tr>
<tr>
<td></td>
<td>yes: Enables automatic refresh. When enabled, the certificate and password</td>
</tr>
<tr>
<td></td>
<td>are regenerated if either is in danger of expiration (less than half the</td>
</tr>
<tr>
<td></td>
<td>time left).</td>
</tr>
<tr>
<td></td>
<td>no: Turns off automatic certificate and password refresh.</td>
</tr>
<tr>
<td></td>
<td>This value is created and the value is set by using one of these utilities:</td>
</tr>
<tr>
<td></td>
<td>• For <code>ivmgrd.conf</code>, it is set by <code>mgrsslcfg</code>.</td>
</tr>
<tr>
<td></td>
<td>• For <code>pd.conf</code>, it is set by <code>basslcfg</code>.</td>
</tr>
<tr>
<td></td>
<td>• For all others, it is set by <code>svrsslcfg</code>.</td>
</tr>
<tr>
<td></td>
<td>This stanza entry is required.</td>
</tr>
<tr>
<td></td>
<td>Example: <code>ssl-auto-refresh = yes</code></td>
</tr>
<tr>
<td><code>ssl-local-domain</code></td>
<td>The name of the local domain. The server runs on this domain. If this value</td>
</tr>
<tr>
<td></td>
<td>is not in the configuration file, then operations that rely on its presence</td>
</tr>
<tr>
<td></td>
<td>will fail.</td>
</tr>
<tr>
<td></td>
<td>The <code>domain_name</code> value is an alphanumeric, case-sensitive string. String</td>
</tr>
<tr>
<td></td>
<td>values are expected to be characters that are part of the local code set.</td>
</tr>
<tr>
<td></td>
<td>For the policy proxy server (<code>pdmgrproxyd</code>) to span domains, this value</td>
</tr>
<tr>
<td></td>
<td>should be the management domain.</td>
</tr>
<tr>
<td></td>
<td>The <code>domain_name</code> value is created during configuration but can be changed</td>
</tr>
<tr>
<td></td>
<td>by using one of these utilities:</td>
</tr>
<tr>
<td></td>
<td>• For <code>ivmgrd.conf</code>, it is changed by using <code>mgrsslcfg</code>.</td>
</tr>
<tr>
<td></td>
<td>• For <code>pd.conf</code>, it is changed by using <code>basslcfg</code>.</td>
</tr>
<tr>
<td></td>
<td>• For all others, it is changed by using <code>svrsslcfg</code>.</td>
</tr>
<tr>
<td></td>
<td>This stanza entry is required.</td>
</tr>
<tr>
<td></td>
<td>Default value for all servers: <code>Default</code></td>
</tr>
<tr>
<td></td>
<td>Example: <code>ssl-local-domain = Default</code></td>
</tr>
<tr>
<td><code>ssl-authn-type</code></td>
<td>Type of authentication.</td>
</tr>
<tr>
<td></td>
<td>This value is created and the value is set during configuration for the</td>
</tr>
<tr>
<td></td>
<td>authentication server <code>pdacld</code> and the policy proxy server <code>pdmgrproxyd</code>.</td>
</tr>
<tr>
<td></td>
<td>However, this stanza entry is not used for the policy server <code>pdmgrd</code>.</td>
</tr>
<tr>
<td></td>
<td>This stanza entry is required only for the authentication and policy proxy</td>
</tr>
<tr>
<td></td>
<td>server.</td>
</tr>
<tr>
<td></td>
<td>Default value for all servers: <code>certificate</code></td>
</tr>
<tr>
<td></td>
<td>Example: <code>ssl-authn-type = certificate</code></td>
</tr>
</tbody>
</table>
ssl-listening-port = (0|port)

TCP port to listen on for incoming requests.

Valid values include:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disables listening. The value is specified during configuration by using the svrsslcfg utility.</td>
</tr>
<tr>
<td>port</td>
<td>Enables listening at the specified port number. The valid range for port is any positive number that is allowed by TCP/IP and is not currently being used by another application.</td>
</tr>
</tbody>
</table>

There is no one default value because the configuration programs for each daemon specifies its own default value. For example, when configuring the policy proxy server, the user is prompted for a port, with 8139 as the default. This value is then used in the call to the SSL configuration utility.

Default value is 0, if not specified during configuration. Otherwise, the value is server-dependent. The policy server pdmgrd does not use this stanza entry.

This stanza entry is required, except for pdmgrd.

Example: ssl-listening-port = 8139
[ssl] stanza for ldap.conf

The ldap.conf configuration file defines the SSL configuration settings for the LDAP server. The stanza entries for configuring SSL settings are located in the [ssl] stanza of this configuration files:

- ldap.conf for the LDAP server

### [ssl] stanza

```
ssl-local-domain = {Default|domain_name}
```

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssl-local-domain</td>
<td>The name of the local domain. The server runs on this domain. If this value is not in the configuration file, then operations that rely on its presence will fail.</td>
</tr>
<tr>
<td></td>
<td>The <code>domain_name</code> value is an alphanumerical, case-sensitive string. The domain name value is created and set by using the <code>svrsslCfg</code> utility:</td>
</tr>
<tr>
<td></td>
<td>This stanza entry is required.</td>
</tr>
<tr>
<td></td>
<td>There is no default value. The name is set by the configuration program.</td>
</tr>
<tr>
<td></td>
<td>Example: <code>ssl-local-domain = Default</code></td>
</tr>
</tbody>
</table>

---

Appendix A. Server configuration file reference 309
[uraf-registry] stanza

A User Registry Adapter Framework (URAF) stanza is required when the configured registry type is not LDAP. The stanza entries for configuring URAF-based registry settings for the server are located in the [uraf-registry] stanza of these configuration files:

- ivmgrd.conf for the policy server **pdmgrd**
- ivacld.conf for the authorization server **pdacld**
- pdmgrproxyd.conf for the policy proxy server **pdmgrproxyd**
- Your resource managers' configuration file for configured registry types that are not LDAP

In addition, you can set additional stanza entries in the [uraf-registry] stanza of the activedir.conf, activedir_ldap.conf, or domino.conf configuration files. The configuration file that is used depends on the type of URAF user registry that you configure.

The aznAPI.conf configuration file is provided with Tivoli Access Manager as a sample file for creating your own resource manager configuration file. Developers of service plug-ins should provide the standard functions. Before implementing service plug-ins, read and thoroughly understand the concepts discussed in the IBM Tivoli Access Manager for e-business Authorization C API Developer Reference.

Most information in this stanza is filled in during configuration. The exception is the cache-related items that must be manually updated by the Tivoli Access Manager administrator. The cache-mode, cache-size, and cache-lifetime stanza entries do not appear in ivmgrd.conf because the policy server’s object should not be cached.

**Note:** Do not place the following stanza entries in the [uraf-registry] stanza of the activedir.conf, activedir_ldap.conf, or domino.conf configuration files.

```plaintext
uraf-registry-config = C:\PROGRA~1\Tivoli\POLICY~1\etc\activedir.conf
bind-id = ivmgrd-master
```
### [uraf-registry] stanza

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uraf-registry-config =</td>
<td>File name and location of the URAF registry configuration file for Tivoli Access Manager. The <strong>fully_qualified_path</strong> value represents an alphanumeric string. String values are expected to be characters that are part of the local code set. The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows, file names cannot have these characters: a backward slash (<code>\</code>), a colon (<code>:</code>), a question mark (<code>?</code>), or double quotation marks (<code>&quot;</code>). For UNIX, path and file names are case sensitive. This stanza entry is required when the configured registry type is not LDAP. The default value is server-specific. It is generated; but it can be changed. The default URAF registry configuration files can be one of the following:   * domino.conf  * activedir.conf  * activedir_ldap.conf   Windows example using IBM Domino as the user registry from a Windows client: uraf-registry-config = C:\Program Files\Tivoli\Policy Director\etc\domino.conf   Windows example for using Microsoft Active Directory user registry for platforms other than Windows 2000: uraf-registry-config = c:\Program files\tivoli\Policy Director\etc\activedir_ldap.conf   Example using Microsoft Active Directory as the registry from a UNIX client: uraf-registry-config = /opt/PolicyDirector/etc/activedir_ldap.conf</td>
</tr>
</tbody>
</table>

### bind-id = server_id

Server administrator or user login identity that is used to bind (sign on) to the registry server. Only the server uses this ID. If the ID belongs to a user rather than an administrator, the user must have privileges to update and modify data in the user registry. For IBM Lotus Domino registry, a Lotus Notes ID file provides the bind ID equivalent. **Note:** This value is generated during configuration; do not change it.

The **server_id** is an alphanumeric, case-insensitive string. String values are expected to be characters that are part of the local code set.

The minimum and maximum lengths of the ID, if there are limits, are imposed by the underlying registry. For Active Directory the maximum length is 256 alphanumeric characters.

This stanza entry is required when the configured registry type is not LDAP. The default value is server-specific.

Example: bind-id = MySvrAdminID
**cache-mode = { enabled | disabled }**

Mode for caching that represents the cache being either turned on or turned off. **Note:** This stanza entry does not appear in the `ivmgrd.conf` configuration file because you do not want policy server object to be cached.

Valid values include:

**enabled**

Turns the cache on. You would enable the cache mode to improve the performance of repetitive Read actions on a specified object, such as: login performance that is done more than once a day. Performance for Write actions would not be improved.

**disabled**

Turns the cache off. You would disable the cache mode for better security. Caching opens a small window for users to go from server to server in order to bypass the maximum number of failed login attempts.

This stanza entry is optional. This stanza entry is normally provided for all Tivoli Access Manager servers, except for the policy server `pdmgrd`.

Default value: enabled

Example: `cache-mode = enabled`

**cache-lifetime = number_seconds**

Number of seconds that the objects are allowed to stay in the cache.

Valid values include:

`number_seconds`

The timeout specified in number of seconds. Use a number within the range of 1 to 86400.

**Note:** This stanza entry does not appear in the `ivmgrd.conf` configuration file because you do not want the policy server object to be cached.

If `cache-mode = enabled` and this stanza entry is not used, the default value of 30 seconds will be used.

For performance tuning, the longer the time specified, the longer the repetitive Read advantage is held. A smaller number of seconds negates the cache advantage for user-initiated Reads.

This stanza entry is optional.

Default value: 30 seconds.

Example: `cache-lifetime = 63200`
cache-size = \{number_objects|object type:cache count value\}

Maximum number of objects for a particular type of object that can be in the cache at one time without hash table collisions. Or, if it is not numeric, it is a list of one or more object types and their cache count values.

**Note:** This stanza entry does not appear in the ivmgrd.conf configuration file because you do not want the policy server object to be cached.

If cache-mode = enabled and this stanza entry is not used, the default value for cache size will be used.

Valid values include:

- **number_objects**
  
  Maximum number of objects must be a prime number for the cache count values. Range value is from 3 to a maximum number that is logical for the task and that does not affect performance. Non-prime numbers are automatically rounded up to the next higher prime number. If the number fails, the default value will be used.

- **object type:cache count value**
  
  List of one or more object types and their cache count values.

  Examples:

  ```
  cache-size = user:251;group:251;resgroup:251;resource:251;rescreds:251;
  rescreds:251;
  ```

  or

  ```
  cache-size = user:251;group:251;
  ```

  The second example sets the user and group cache sizes to 251 and does not use any cache for the others.

Performance tuning depends on how much memory space is dedicated to a cache or how many objects you typically have repetitive Read actions on (such as how many users you have logging in a day). For example, a setting of 251 might not be good if you have 1000 users logging in and out several times a day. However, if only 200 of those users log in and out repetitively during the day, 251 might work well.

This stanza entry is optional.

The default value is server-specific.

Example: cache-size = 251
**[uraf-registry] stanza for domino.conf**

The stanza entries for configuring an IBM Lotus Domino server as the user registry are located in the **[uraf-registry] stanza** of this configuration file:

* domino.conf to configure IBM Lotus Domino as the user registry server

<table>
<thead>
<tr>
<th><strong>[uraf-registry] stanza</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>**enabled = {yes</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Indication of whether Domino is being used as the user registry.</td>
</tr>
<tr>
<td>Valid values include:</td>
</tr>
<tr>
<td><strong>yes</strong></td>
</tr>
<tr>
<td>Indicates Domino is the user registry.</td>
</tr>
<tr>
<td><strong>no</strong></td>
</tr>
<tr>
<td>Indicates Domino is not the user registry. Anything other than yes, including a blank value, is interpreted as no.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>This stanza entry is required.</td>
</tr>
<tr>
<td>Default value: no</td>
</tr>
<tr>
<td>Example: enabled = yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>server = server_name</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name of the IBM Lotus Domino server.</td>
</tr>
<tr>
<td>The <em>server_name</em> value represents an alphanumeric, case-insensitive string. String values are expected to be characters that are part of the local code set. The minimum and maximum lengths of the name are imposed by the underlying registry.</td>
</tr>
<tr>
<td>This stanza entry is required when enabled = yes.</td>
</tr>
<tr>
<td>There is no default value.</td>
</tr>
<tr>
<td>Example: server = grizzly/Austin/IBM</td>
</tr>
<tr>
<td>Where grizzly is the Domino server machine host name and the remainder is the Domino domain name.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>NAB = names.nsf</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Lotus Domino Name and Address Book (NAB) database. The <em>names.nsf</em> file name conforms to the underlying operating system file naming conventions of the Domino server.</td>
</tr>
<tr>
<td>This stanza entry is required when enabled = yes.</td>
</tr>
<tr>
<td>The <em>names.nsf</em> database is set at configuration time and cannot be changed. The file name extension must always be <em>.nsf</em>.</td>
</tr>
<tr>
<td>Default value: names.nsf</td>
</tr>
<tr>
<td>Example: NAB = names.nsf</td>
</tr>
<tr>
<td>PDM = nsf_filename</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>Tivoli Access Manager meta-data database.</td>
</tr>
<tr>
<td>The nsf_filename represents a Domino database file name. The file name conforms to the underlying operating system file naming conventions of the Domino server. The database is created on the Domino server during configuration and cannot be changed. The recommended file name extension is .nsf.</td>
</tr>
<tr>
<td>This stanza entry is required when enabled = yes.</td>
</tr>
<tr>
<td>Default value: PDMdata.nsf</td>
</tr>
<tr>
<td>Example: PDM = PDMdata.nsf</td>
</tr>
</tbody>
</table>
## [uraf-registry] stanza for activedir.conf

The stanza entries for configuring Microsoft Active Directory server as the user registry are located in the [uraf-registry] stanza of this configuration file:

- activedir.conf to configure Microsoft Active Directory as the user registry.

### [uraf-registry] stanza

| enabled = {yes|no} |
|-------------------|
| Indication of whether Active Directory is being used as the user registry. |
| Valid values include: |
| yes Indicates Active Directory is the user registry. |
| no Indicates Active Directory is not the user registry. Anything other than yes, including a blank value, is interpreted as no. |

This stanza entry is required when your user registry is Microsoft Active Directory.

Default value: no

Example: enabled = yes

### multi-domain = {true|admd|ammd|false}

Indication of whether the domain is a single-domain or multi-domain configuration. Selection is made at the time of runtime configuration of Tivoli Access Manager.

**Note:** This stanza entry is set during configuration; do not edit it.

Valid values include:

<table>
<thead>
<tr>
<th>true</th>
<th>admd</th>
</tr>
</thead>
<tbody>
<tr>
<td>For Tivoli Access Manager single domain with Active Directory multiple domains.</td>
<td></td>
</tr>
<tr>
<td>ammd</td>
<td>For multiple Tivoli Access Manager domains, implying multiple Active Directory domains.</td>
</tr>
<tr>
<td>false</td>
<td>For a single Active Directory domain.</td>
</tr>
</tbody>
</table>

This stanza entry is required when your user registry is Microsoft Active Directory.

There is no default value.

Example: multi-domain = true

### hostname = host_name

Active Directory DNS host name. Examples of valid host names:

- mycomputer.city.company.com
- mycomputer

The host_name value is automatically filled in during runtime configuration. The host name value is an alphanumeric, case-insensitive string. String values are expected to be characters that are part of the local code set. The dot (.) cannot be the last character of the host name. The maximum string length for the Active Directory user registry is 256 alphanumeric characters.

This stanza entry is required when your user registry is Microsoft Active Directory.

There is no default value.

Example: hostname = adserver.tivoli.com
**domain = root_domain_name**

Active Directory root (primary) domain. This name is domain-dependent, based on what you select during runtime configuration of Tivoli Access Manager.

The *root_domain_name* is an alphanumeric, case-sensitive string. String values are expected to be characters that are part of the local code set. The maximum length for the domain name is user registry dependent. For Active Directory that maximum length is 256 alphanumeric characters.

This stanza entry is required when your user registry is Microsoft Active Directory and when `multi-domain = {true|admd}`.

There is no default value.

Example: `domain = dc=tivoli,dc=com`

---

**useEncryption = {true|false}**

Indication of whether encryption communication to Active Directory is being used. This value is specified at runtime configuration of Tivoli Access Manager.

Valid values include:
- **true** Enables encryption communication.
- **false** Disables encryption communication.

This stanza entry is required when your user registry is Microsoft Active Directory.

There is no default value.

Example: `useEncryption = false`

---

**dnforpd = ad_dn**

Distinguished name that is used by Active Directory to store Tivoli Access Manager data. **Note:** This stanza entry is set during configuration; do not change it.

The *ad_dn* value is an alphanumeric, case-insensitive string. String values are expected to be characters that are part of the local code set. The minimum and maximum lengths of the ID, if there are limits, are imposed by the underlying registry. For Active Directory the maximum length is 256 alphanumeric characters.

This stanza entry is required when your user registry is Microsoft Active Directory.

There is no default value.

Example: `dnforpd = dc=child2,dc=com`
[uraf-registry] stanza for activedir_ldap.conf

When you use an LDAP client to retrieve data for the Active Directory user registry that the Tivoli Access Manager server is configured to, you must have the server configuration file activedir_ldap.conf. Use this configuration file to customize the operation of each Active Directory registry server.

The stanza entries for configuring the Microsoft Active Directory as the user registry on a Tivoli Access Manager server are located in the [uraf-registry] stanza of this configuration file:

* activedir_ldap.conf

<table>
<thead>
<tr>
<th>[uraf-registry] stanza</th>
</tr>
</thead>
<tbody>
<tr>
<td>enabled = {yes</td>
</tr>
<tr>
<td>Indication of whether Active Directory is being used as the user registry. Valid values include:</td>
</tr>
<tr>
<td>yes  Indicates Active Directory is the user registry.</td>
</tr>
<tr>
<td>no   Indicates Active Directory is not the user registry. Anything other than yes, including a blank value, is interpreted as no.</td>
</tr>
<tr>
<td>This stanza entry is required when your user registry is Microsoft Active Directory.</td>
</tr>
<tr>
<td>Default value: no</td>
</tr>
<tr>
<td>Example: enabled = yes</td>
</tr>
</tbody>
</table>

| multi-domain = {true|admd|ammd|false} |
|---------------------------------------|
| Indication of whether the domain is a single-domain or multi-domain configuration. Selection is made at the time of runtime configuration of Tivoli Access Manager. |
| Note: This stanza entry is set during configuration; do not edit it. |
| Valid values include:                 |
| true|admd For Tivoli Access Manager single domain with Active Directory multiple domains. |
| ammd For multiple Tivoli Access Manager domains, implying multiple Active Directory domains. |
| false For a single Active Directory domain. |
| This stanza entry is required when your user registry is Microsoft Active Directory. |
| There is no default value.            |
| Example: multi-domain = true          |

| UseSSL = {yes|no}                      |
|---------------------------------------|
| Indication of whether to use SSL.     |
| Value values:                         |
| yes  Specifies that you want to use SSL. |
| no   Specifies that you do not want to use SSL. |
| This stanza entry is required.        |
| Default value: yes                    |
| Example: useSSL = no                  |
**ssl-keyfile = ldap-ssl-key-filename**

SSL key file name and location. Use the SSL key file to handle certificates that are used in LDAP communication. The file type can be anything but the extension is usually `.kdb`.

The certificate files in a directory need to be accessible to the server user (or all users). Make sure that server user (for example, ivmgr) or all users have permission to access the `.kdb` file and the folder that contains the `.kdb` file.

The file name and location value represents an alphanumeric, case-insensitive string. String values are expected to be characters that are part of the local code set. The set of characters permitted in a file name can be determined by the file system and by the local code set. For Windows, file names cannot have these characters: a backward slash (`\`), a colon (`:`), a question mark (`?`), or double quotation marks (`"`). The maximum string length for the Active Directory user registry is 256 alphanumeric characters.

This stanza entry is required only when `ssl-enabled = yes`.

Default value is server dependent.

Default value for Windows:

```
pd_install_dir\keytab\server_name.kdb
```

Default value for UNIX:

```
/opt/PolicyDirector/keytab/server_name.kdb
```

Example for UNIX for the policy server `pdmgrd`: `ssl-keyfile = /opt/PolicyDirector/keytab/ivmgrd.kdb`

**ssl-keyfile-label = key_label**

Specifies the key label that is used to identify the client certificate that is presented to the LDAP server. It is the key label of the client personal certificate within the SSL key file.

The `key_label` is an alphanumeric, case-insensitive string. String values are expected to be characters that are part of the local code set. The minimum and maximum lengths of the ID, if there are limits, are imposed by the underlying registry. The key label must be enclosed in double quotation marks.

This stanza entry is required when the LDAP server is configured to perform client authentication.

There is no default value.

Example: `ssl-keyfile-label = "PDLAP"`
### ssl-keyfile-pwd = ldap-ssl-keyfile-password

Password to access the SSL key file. The password associated with the default SSL keyfile is `key4ssl`.

This stanza entry is required only if enabled = yes.

There is no default value.

Example: `ssl-keyfile-pwd = key4ssl`

### primary-domain = primary_domain_name

Active Directory primary domain host name, and zero or more replica host names. Only one primary domain entry is allowed. This name is domain-dependent, based on what you select during runtime configuration of Tivoli Access Manager.

The `primary_domain_name` is an alphanumeric, case-sensitive string. String values are expected to be characters that are part of the local code set. The maximum length for the domain name is user registry dependent. For Active Directory that maximum length is 256 alphanumeric characters.

For the Active Directory multi-domain configuration, the primary domain entry must contain the root domain information.

For the Active Directory single domain configuration, either `primary-domain =` or `domain =` stanza entries can be used for the domain information.

The syntax is:

```
primary-domain = nnn:hhh[:rrr1[:rrr2[...]]]
```

Where:

- **nnn**: The primary domain name. The name format can be either `ibm.com` or `dc=ibm,dc=com`.
- **hhh**: The primary domain host name or IP address.
- **rrr**: The primary domain replica host name or IP address.

Square brackets ([ ]) show entries that are optional and the required colon (:) acts as a separator.

This stanza entry is required.

There is no default value.

Example (typed on one line) without spaces:

```
primary-domain = dc=ibm,dc=com:adprim.ibm.com:adprimreplical.ibm.com
```
<table>
<thead>
<tr>
<th><strong>domain = secondary_domain_name</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name of Active Directory secondary or child domain host name. This host name is in the same forest as the root domain, its host name, and zero or more replica host names. This name is domain-dependent, based on what you select during runtime configuration of Tivoli Access Manager.</td>
</tr>
<tr>
<td>The secondary_domain_name is an alphanumeric, case-sensitive string. String values are expected to be characters that are part of the local code set. The maximum length for the domain name is user registry dependent. For Active Directory that maximum length is 256 alphanumeric characters.</td>
</tr>
<tr>
<td>For the Active Directory single domain configuration, either primary-domain= or domain= can be used to enter the domain name information.</td>
</tr>
<tr>
<td>For the Active Directory multiple domain configuration, multiple domain name entries are allowed.</td>
</tr>
<tr>
<td>The syntax is:</td>
</tr>
<tr>
<td>domain = nnn:hhh[:rrr1[:rrr2[:...]]]</td>
</tr>
<tr>
<td>where:</td>
</tr>
<tr>
<td>nnn</td>
</tr>
<tr>
<td>hhh</td>
</tr>
<tr>
<td>rrr</td>
</tr>
<tr>
<td>Square brackets ([ ]) show entries that are optional and the required colon (:) acts as a separator.</td>
</tr>
<tr>
<td>This stanza entry is required when your user registry is Microsoft Active Directory and when multi-domain = {true</td>
</tr>
<tr>
<td>There is no default value.</td>
</tr>
<tr>
<td>Example (typed on one line) without spaces:</td>
</tr>
</tbody>
</table>

<p>| <strong>ldap-client-timeout = {0|number_seconds}</strong> |
|-----------------------------------------------|
| Amount of time that is allowed for to LDAP simple bind and LDAP searches before the LDAP client is considered to be down. |
| Valid values include: |
| 0 | Unlimited amount of time, in seconds, allowed only for synchronous operations. |
| number_seconds | Amount of time, in seconds, allowed for asynchronous operations. The number of seconds is specified as a positive integer whole number. The suggested range is between 240 to 900 seconds. |
| This stanza entry is required. |
| Default value: 0 |
| Example: ldap-client-timeout = 520 |</p>
<table>
<thead>
<tr>
<th>** dnforpd = ad_dn **</th>
</tr>
</thead>
</table>
| Distinguished name that is used by Active Directory to store Tivoli Access Manager data. **Note:** This stanza entry value is set during configuration; do not change it.  
The `ad_dn` value is an alphanumeric, case-insensitive string. String values are expected to be characters that are part of the local code set. The minimum and maximum lengths of the ID, if there are limits, are imposed by the underlying registry. For Active Directory the maximum length is 256 alphanumeric characters.  
This stanza entry is required when your user registry is Microsoft Active Directory.

There is no default value.

Example: `dnforpd = dc=child2,dc=com`
[xmladi-attribute-definitions] stanza

The stanza entries for configuring the Access Decision Information eXtensible Markup Language (ADI XML) document attribute definitions are located in the [xmladi-attribute-definitions] stanza. This stanza can be found or placed into any of the Tivoli Access Manager configuration files, except for the pd.conf configuration file.

The aznAPI.conf configuration file is provided with Tivoli Access Manager as a sample file for creating your own resource manager configuration file. Developers of service plug-ins should provide the standard functions. Before implementing service plug-ins, read and thoroughly understand the concepts discussed in the IBM Tivoli Access Manager for e-business Authorization C API Developer Reference.

<table>
<thead>
<tr>
<th>AttributeName = &quot;AttributeValue&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADI XML document attribute definitions that are inserted into the XML ADI element start tag to enable attributes to be defined for the entire XML ADI document and for all ADI defined in the XML ADI document.</td>
</tr>
</tbody>
</table>

The ADI XML model requires the XML document to contain the following top-level XML element into which all target ADI for a particular rule evaluation is inserted. The XMLADI element is created automatically as part of the rule evaluation process by the authorization engine.

```xml
<XMLADI>
<!-- XML formatted ADI are inserted here. -->
</XMLADI>
```

This stanza entry is required.

Examples of two attributes:

```xml
xmlns:myNS = "http://myURI.mycompany.com"
appID = "Jupiter - Account Management Web Portal Server #1."
```

The attribute value must be enclosed in either double or single quotation marks.

The XMLADI element start tag built from these definitions is:

```xml
<XMLADI xmlns:myNS="http://myURI.mycompany.com"
         appID="Jupiter - Account Management Web Portal Server #1."
></XMLADI>
```

For more information, see Chapter 10, “Authorization rules management,” on page 101.
Appendix B. 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:
   
   pdadmin user create myuser username/locinfo test testpwd
   
   instead of using this one:
   
   pdadmin 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 11 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 11. 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>
Table 11. 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 description, POP description</td>
<td>unlimited</td>
<td>unlimited</td>
<td>unlimited</td>
<td></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.
Appendix C. Administration command line and Web Portal Manager equivalents

This appendix shows the mapping that exists between the administration pdadmin command line interface (CLI) and Web Portal Manager.

Information about the pdadmin command line interface can be found in the IBM Tivoli Access Manager for e-business Command Reference.

Table 12. Mapping between administration CLI and Web Portal Manager

<table>
<thead>
<tr>
<th>pdadmin CLI</th>
<th>Web Portal Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>acl attach object_name acl_name</td>
<td>ACL → List ACL → click POP name → Attach tab → Attach</td>
</tr>
<tr>
<td>acl create acl_name</td>
<td>ACL → Create ACL</td>
</tr>
<tr>
<td>acl delete acl_name</td>
<td>ACL → List ACL → select ACL names → Delete</td>
</tr>
<tr>
<td>acl detach object_name</td>
<td>ACL → List ACL → click ACL name → Attach tab → select object names → Detach</td>
</tr>
<tr>
<td>acl find acl_name</td>
<td>ACL → List ACL → click ACL name → Attach tab</td>
</tr>
<tr>
<td>acl list</td>
<td>ACL → List ACL</td>
</tr>
<tr>
<td>acl list acl_name attribute</td>
<td>ACL → List ACL → click ACL name → Extended Attribute tab</td>
</tr>
<tr>
<td>acl modify acl_name delete attribute attribute_name</td>
<td>ACL → List ACL → select ACL name → Extended Attribute tab → select attribute → Delete</td>
</tr>
<tr>
<td>acl modify acl_name delete attribute attribute_name attribute_value</td>
<td>ACL → List ACL → click ACL name → Extended Attribute tab → select attributes → Delete</td>
</tr>
<tr>
<td>acl modify acl_name description</td>
<td>ACL → List ACL → click ACL name → modify Description → Set</td>
</tr>
<tr>
<td>acl modify acl_name remove any-other</td>
<td>ACL → List ACL → click ACL name → select Any-other ACL Entry → Delete</td>
</tr>
<tr>
<td>acl modify acl_name remove group group_name</td>
<td>ACL → List ACL → click ACL name → select Group ACL Entry → Delete</td>
</tr>
<tr>
<td>acl modify acl_name remove unauthenticated</td>
<td>ACL → List ACL → click ACL name → select Unauthenticated ACL Entry → Delete</td>
</tr>
<tr>
<td>acl modify acl_name remove user user_name</td>
<td>ACL → List ACL → click ACL name → select User ACL Entry → Delete</td>
</tr>
<tr>
<td>acl modify acl_name set any-other perms</td>
<td>ACL → List ACL → click ACL name → click Any-other Permissions → select permissions → Apply</td>
</tr>
<tr>
<td>acl modify acl_name set attribute attribute_name attribute_value</td>
<td>ACL → List ACL → click ACL name → Extended Attribute tab → Create</td>
</tr>
<tr>
<td>acl modify acl_name set group group_name perms</td>
<td>ACL → List ACL → click ACL name → Create → choose Entry Type Group → specify name of group → select permissions → Apply</td>
</tr>
<tr>
<td>pdadmin CLI</td>
<td>Web Portal Manager</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>acl modify acl_name set unauthenticated perms</td>
<td>ACL → List ACL → click ACL name → Create → choose Entry Type Unauthenticated → select permissions → Apply</td>
</tr>
<tr>
<td>acl modify acl_name set user user_name perms</td>
<td>ACL → List ACL → click ACL name → Create → choose Entry Type User → specify name of User → select permissions → Apply</td>
</tr>
<tr>
<td>acl show acl_name</td>
<td>ACL → List ACL → click ACL name</td>
</tr>
<tr>
<td>acl show acl_name attribute attribute_name</td>
<td>ACL → List ACL → click ACL name → Extended Attribute tab</td>
</tr>
<tr>
<td>acl show any-other</td>
<td>ACL → List ACL → click ACL name</td>
</tr>
<tr>
<td>action create name description action_type</td>
<td>ACL → List Action Groups → click Action Group → Create → fill in form → Create</td>
</tr>
<tr>
<td>action create name description action_type action_group_name</td>
<td>ACL → List Action Groups → click Action Group → Create → fill in form → Create</td>
</tr>
<tr>
<td>action delete name</td>
<td>ACL → List Action Groups → select primary action group → select actions → Delete</td>
</tr>
<tr>
<td>action delete name action_group_name</td>
<td>ACL → List Action Groups → click Action Group → select actions → Delete</td>
</tr>
<tr>
<td>action group create action_group_name</td>
<td>ACL → Create Action Group</td>
</tr>
<tr>
<td>action group delete action_group_name</td>
<td>ACL → List Action Groups → select action groups → Delete</td>
</tr>
<tr>
<td>action group list</td>
<td>ACL → List Action Groups</td>
</tr>
<tr>
<td>action list</td>
<td>ACL → List Action Groups → click primary action group</td>
</tr>
<tr>
<td>action list action_group_name</td>
<td>ACL → List Action Groups → click Action Group</td>
</tr>
<tr>
<td>admin show configuration</td>
<td>Not supported</td>
</tr>
<tr>
<td>authzrule attach protoobjid ruleid</td>
<td>AuthzRule → List AuthzRule → click AuthzRule name → Attach tab → Attach</td>
</tr>
<tr>
<td>authzrule create ruleid [--rulefile filename</td>
<td>ruletext] [-desc description] [-failreason failreason]</td>
</tr>
<tr>
<td>authzrule delete ruleid</td>
<td>AuthzRule → List AuthzRule → select AuthzRule names → Delete</td>
</tr>
<tr>
<td>authzrule detach protoobjid</td>
<td>AuthzRule → List AuthzRule → click AuthzRule name → Attach tab → Detach</td>
</tr>
<tr>
<td>authzrule find ruleid</td>
<td>AuthzRule → List AuthzRule → click AuthzRule name → Attach tab</td>
</tr>
<tr>
<td>authzrule list</td>
<td>AuthzRule → List AuthzRule</td>
</tr>
<tr>
<td>authzrule modify ruleid [--rulefile filename</td>
<td>ruletext] [description description] [failreason failreason]</td>
</tr>
<tr>
<td>authzrule show ruleid</td>
<td>AuthzRule → List AuthzRule → click AuthzRule name</td>
</tr>
<tr>
<td>pdadmin CLI</td>
<td>Web Portal Manager</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>config modify svrpassword config_file password</td>
<td>Not supported</td>
</tr>
<tr>
<td>config modify keyvalue set[-obfuscate] config_file stanza key value</td>
<td>Not supported</td>
</tr>
<tr>
<td>config modify keyvalue append [-obfuscate] config_file stanza key value</td>
<td>Not supported</td>
</tr>
<tr>
<td>config modify keyvalue remove config_file stanza key value</td>
<td>Not supported</td>
</tr>
<tr>
<td>config modify keyvalue remove config_file stanza key</td>
<td>Not supported</td>
</tr>
<tr>
<td>config show config_file stanza key</td>
<td>Not supported</td>
</tr>
<tr>
<td>pdadmin context show</td>
<td>Not supported</td>
</tr>
<tr>
<td>domain create domain domain_admin_id domain_admin_id domain_admin_password [-desc description]</td>
<td>Secure Domain → Create Secure Domain</td>
</tr>
<tr>
<td>domain delete domain [-registry]</td>
<td>Secure Domain → List Secure Domain select Secure Domain names → Delete</td>
</tr>
<tr>
<td>domain list</td>
<td>Secure Domain → List Secure Domain</td>
</tr>
<tr>
<td>domain modify domain description</td>
<td>Secure Domain → List Secure Domain click Secure Domain name → modify description → Apply</td>
</tr>
<tr>
<td>domain show domain</td>
<td>Secure Domain → List Secure Domain click Secure Domain name</td>
</tr>
<tr>
<td>errtext error_number</td>
<td>Not supported</td>
</tr>
<tr>
<td>exit</td>
<td>Not supported</td>
</tr>
<tr>
<td>group create group_name dn cn [group_container]</td>
<td>Group → Create Group</td>
</tr>
<tr>
<td>group delete [-registry] group_name</td>
<td>Group → Search Groups enter pattern and maximum results → Search select group names → Delete</td>
</tr>
<tr>
<td>group import group_name dn [group_container]</td>
<td>Group → Import Group</td>
</tr>
<tr>
<td>group list pattern max_return</td>
<td>Group → Search Groups enter pattern and maximum results → Search</td>
</tr>
<tr>
<td>group list-dn pattern max_return</td>
<td>Not supported</td>
</tr>
<tr>
<td>group modify group_name add (user_name1 user_name2 ... )</td>
<td>Group → Search Groups enter pattern and maximum results → Search click group name → Members tab → Add</td>
</tr>
<tr>
<td>group modify group_name description description</td>
<td>Group → Search Groups enter pattern and maximum results → Search click group name → enter Description → Apply</td>
</tr>
<tr>
<td>group modify group_name remove (user_name1 user_name2 ...)</td>
<td>Group → Search Groups enter pattern and maximum results → Search click group name → Members tab → select user names → Remove</td>
</tr>
<tr>
<td>group show group_name</td>
<td>Group → Search Groups enter pattern and maximum results → Search click group name</td>
</tr>
<tr>
<td>pdadmin CLI</td>
<td>Web Portal Manager</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------------------------------------------------------------</td>
</tr>
<tr>
<td>group show-dn dn</td>
<td>Not supported</td>
</tr>
<tr>
<td>group show-members group_name</td>
<td>Group → Search Groups → enter pattern and maximum results → Search → click group name → Members tab</td>
</tr>
<tr>
<td>help {topic</td>
<td>command}</td>
</tr>
<tr>
<td>login -a admin_id -p password [-domain][-m]</td>
<td>Not supported</td>
</tr>
<tr>
<td>login -l</td>
<td>Not supported</td>
</tr>
<tr>
<td>logout</td>
<td>Not supported</td>
</tr>
<tr>
<td>object access object_name permissions</td>
<td>Not supported</td>
</tr>
</tbody>
</table>
| object create object_name description type ispolicyattachable {yes|no} | Object Space → Create Object  
The type field is not supported.  
You can select the Can Policy be attached to this object check box on the Protected Object Properties window. |
| object delete object_name                      | Object Space → Browse Object Space → expand and click on object name → General tab → Delete |
| object exists object_name                      | Not supported                                               |
| object list                                    | Object Space → Browse Object Space → expand                  |
| object list object_name                        | Object Space → Browse Object Space → expand and click on object name |
| object list object_name attribute              | Object Space → Browse Object Space → expand and click on object name → Extended Attributes tab |
| object listandshow object_name                 | Not supported                                               |
| object modify object_name description description | Object Space → Browse Object Space → expand and click on object name → General tab → Apply |
| object modify object_name delete attribute_name [attribute_value] | Object Space → Browse Object Space → expand and click on object name → Extended Attributes tab → select attribute → Delete |
| object modify object_name set attribute attribute_name attribute_value | Object Space → Browse Object Space → expand and click on object name → Extended Attributes tab → Create |
| object modify object_name set description description | Object Space → Browse Object Space → expand and click on object name → General tab → Apply |
| object modify object_name isPolicyAttachable {yes|no} | Object Space → Browse Object Space → expand and click on object name → General tab → Apply |
| object modify object_name type type            | Not supported                                               |
| object show object_name                        | Object Space → Browse Object Space → expand and click on object name → General tab |
Table 12. Mapping between administration CLI and Web Portal Manager (continued)

<table>
<thead>
<tr>
<th>pdadmin CLI</th>
<th>Web Portal Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>object show object_name attribute</td>
<td>Object Space → Browse Object Space → expand and click on object name → Extended Attributes tab</td>
</tr>
<tr>
<td>attribute_name</td>
<td></td>
</tr>
<tr>
<td>objectspace create objectspace_name</td>
<td>Object Space → Create Object Space</td>
</tr>
<tr>
<td>objectspace delete objectspace_name</td>
<td>Object Space → Browse Object Space → click object space name → Delete</td>
</tr>
<tr>
<td>objectspace list</td>
<td>Object Space → Browse Object Space</td>
</tr>
<tr>
<td>policy get account-expiry-date</td>
<td>User → Show Global User Policy → Account Expiration Date</td>
</tr>
<tr>
<td>policy get account-expiry-date -user user_name</td>
<td>User → Search Users → enter pattern and maximum results → Search → click user name → Policy tab</td>
</tr>
<tr>
<td>policy get disable-time-interval</td>
<td>User → Show Global User Policy → Disable Time Interval</td>
</tr>
<tr>
<td>policy get disable-time-interval -user user_name</td>
<td>User → Search Users → enter pattern and maximum results → Search → click user name → Policy tab</td>
</tr>
<tr>
<td>policy get max-login-failures</td>
<td>User → Show Global User Policy → Max Login Failures</td>
</tr>
<tr>
<td>policy get max-login-failures -user user_name</td>
<td>User → Search Users → enter pattern and maximum results → Search → click user name → Policy tab</td>
</tr>
<tr>
<td>policy get max-password-age</td>
<td>User → Show Global User Policy → Maximum Password Age</td>
</tr>
<tr>
<td>policy get max-password-age -user user_name</td>
<td>User → Search Users → enter pattern and maximum results → Search → click user name → Policy tab</td>
</tr>
<tr>
<td>policy get max-password-repeated-chars</td>
<td>User → Show Global User Policy → Maximum Password Repeated Characters</td>
</tr>
<tr>
<td>policy get max-password-repeated-chars -user user_name</td>
<td>User → Search Users → enter pattern and maximum results → Search → click user name → Policy tab</td>
</tr>
<tr>
<td>policy get min-password-alphas</td>
<td>User → Show Global User Policy → Minimum Password Alphas</td>
</tr>
<tr>
<td>policy get min-password-alphas -user user_name</td>
<td>User → Search Users → enter pattern and maximum results → Search → click user name → Policy tab</td>
</tr>
<tr>
<td>policy get min-password-length</td>
<td>User → Show Global User Policy → Minimum Password Length</td>
</tr>
<tr>
<td>policy get min-password-length -user user_name</td>
<td>User → Search Users → enter pattern and maximum results → Search → click user name → Policy tab</td>
</tr>
<tr>
<td>policy get min-password-non-alphas</td>
<td>User → Show Global User Policy → Minimum Password Non-Alphas</td>
</tr>
<tr>
<td>policy get min-password-non-alphas -user user_name</td>
<td>User → Search Users → enter pattern and maximum results → Search → click user name → Policy tab</td>
</tr>
</tbody>
</table>
Table 12. Mapping between administration CLI and Web Portal Manager (continued)

<table>
<thead>
<tr>
<th>pdadmin CLI</th>
<th>Web Portal Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>policy get password-spaces</td>
<td>User → Show Global User Policy → Password Spaces Allowed</td>
</tr>
<tr>
<td>policy get password-spaces -user user_name</td>
<td>User → Search Users → enter pattern and maximum results → Search → click user name → Policy tab</td>
</tr>
<tr>
<td>policy get tod-access</td>
<td>User → Show Global User Policy → Time of Day Access</td>
</tr>
<tr>
<td>policy get tod-access -user user_name</td>
<td>User → Search Users → enter pattern and maximum results → Search → click user name → Policy tab</td>
</tr>
<tr>
<td>policy set account-expiry-date [unlimited</td>
<td>absolute_time</td>
</tr>
<tr>
<td>policy set account-expiry-date [unlimited</td>
<td>absolute_time</td>
</tr>
<tr>
<td>policy set disable-time-interval [number</td>
<td>unset</td>
</tr>
<tr>
<td>policy set disable-time-interval [number</td>
<td>unset</td>
</tr>
<tr>
<td>policy set max-login-failures [number</td>
<td>unset]</td>
</tr>
<tr>
<td>policy set max-login-failures [number</td>
<td>unset] -user user_name</td>
</tr>
<tr>
<td>policy set max-password-age [unset</td>
<td>relative_time]</td>
</tr>
<tr>
<td>policy set max-password-age [unset</td>
<td>relative_time] -user user_name</td>
</tr>
<tr>
<td>policy set max-password-repeated-chars [number</td>
<td>unset]</td>
</tr>
<tr>
<td>policy set max-password-repeated-chars [number</td>
<td>unset] -user user_name</td>
</tr>
<tr>
<td>policy set min-password-alphas [number</td>
<td>unset]</td>
</tr>
<tr>
<td>policy set min-password-alphas [number</td>
<td>unset] -user user_name</td>
</tr>
<tr>
<td>policy set min-password-length [number</td>
<td>unset]</td>
</tr>
<tr>
<td>policy set min-password-length [number</td>
<td>unset] -user user_name</td>
</tr>
<tr>
<td>policy set min-password-non-alphas [number</td>
<td>unset]</td>
</tr>
</tbody>
</table>
Table 12. Mapping between administration CLI and Web Portal Manager (continued)

<table>
<thead>
<tr>
<th>pdadmin CLI</th>
<th>Web Portal Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>policy set min-password-non-alphas [number]</td>
<td>User → Search Users → enter pattern and maximum results → Search → click user name → Policy tab</td>
</tr>
<tr>
<td>-user user_name</td>
<td></td>
</tr>
<tr>
<td>policy set password-spaces [yes]no</td>
<td>unset</td>
</tr>
<tr>
<td>-user user_name</td>
<td></td>
</tr>
<tr>
<td>policy set tod-access todaccess_value</td>
<td>User → Show Global User Policy → Time of Day Access → Apply</td>
</tr>
<tr>
<td>-user user_name</td>
<td></td>
</tr>
<tr>
<td>pop attach object_name pop_name</td>
<td>POP → List POP → click POP name → Attach tab → Attach</td>
</tr>
<tr>
<td>pop create pop_name</td>
<td>POP → Create POP</td>
</tr>
<tr>
<td>pop delete pop_name</td>
<td>POP → List POP → select POP names → Delete</td>
</tr>
<tr>
<td>pop detach object_name</td>
<td>POP → List POP → click POP name → Attach tab → select object → Detach</td>
</tr>
<tr>
<td>pop find pop_name</td>
<td>POP → List POP → click POP name → Attach tab</td>
</tr>
<tr>
<td>pop list</td>
<td>POP → List POP</td>
</tr>
<tr>
<td>pop list pop_name</td>
<td>POP → List POP → click POP name</td>
</tr>
<tr>
<td>pop list pop_name attribute</td>
<td>POP → List POP → click POP name → Extended Attributes tab</td>
</tr>
<tr>
<td>pop modify pop_name delete attribute attribute_name</td>
<td>POP → List POP → click POP name → Extended Attributes tab → select attributes → Delete</td>
</tr>
<tr>
<td>pop modify pop_name delete attribute attribute_name attribute_value</td>
<td>POP → List POP → click POP name → Extended Attributes tab → select attributes → Delete</td>
</tr>
<tr>
<td>pop modify pop_name set attribute attribute_value</td>
<td>POP → List POP → click POP name → Extended Attributes tab → Create</td>
</tr>
<tr>
<td>pop modify pop_name set audit-level [all</td>
<td>none</td>
</tr>
<tr>
<td>pop modify pop_name set description description</td>
<td>POP → List POP → click POP name → General tab → Apply</td>
</tr>
<tr>
<td>pop modify pop_name set ipauth add network netmask authentication_level</td>
<td>POP → List POP → click POP name → IP Auth tab → Create → enter the network, net mask, and authentication level → Apply</td>
</tr>
<tr>
<td>pop modify pop_name set ipauth add network netmask forbidden</td>
<td>POP → List POP → click POP name → IP Auth tab → Create → enter the network and net mask, select Forbidden check box → Apply</td>
</tr>
<tr>
<td>pdadmin CLI</td>
<td>Web Portal Manager</td>
</tr>
<tr>
<td>---------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>pop modify pop_name set ipauth anyothernw authentication_level</td>
<td>POP &gt; List POP &gt; click POP name &gt; IP Auth tab &gt; Create &gt; select Any Other Network check box, enter the authentication level &gt; Create</td>
</tr>
<tr>
<td>pop modify pop_name set ipauth anyothernw forbidden</td>
<td>POP &gt; List POP &gt; click POP name &gt; IP Auth tab &gt; Create &gt; select Any Other Network check box &gt; Create</td>
</tr>
<tr>
<td>pop modify pop_name set ipauth remove network netmask</td>
<td>POP &gt; List POP &gt; click POP name &gt; IP Auth tab &gt; select IP auth entries &gt; Delete</td>
</tr>
<tr>
<td>pop modify pop_name set qop {none|integrity|privacy}</td>
<td>POP &gt; List POP &gt; click POP name &gt; General tab &gt; Apply</td>
</tr>
<tr>
<td>pop modify pop_name set tod-access {anyday weekday</td>
<td>day_list}:{anytime</td>
</tr>
<tr>
<td>pop modify pop_name set warning {yes|no}</td>
<td>POP &gt; List POP &gt; click POP name &gt; General tab &gt; Apply</td>
</tr>
<tr>
<td>pop show pop_name</td>
<td>POP &gt; List POP &gt; click POP name</td>
</tr>
<tr>
<td>pop show pop_name attribute</td>
<td>POP &gt; List POP &gt; click POP name &gt; Extended Attributes tab</td>
</tr>
<tr>
<td>quit</td>
<td>Not supported</td>
</tr>
<tr>
<td>rsrc create resource_name</td>
<td>GSO Resource &gt; Create GSO</td>
</tr>
<tr>
<td>rsrc create resource_name -desc description</td>
<td>GSO Resource &gt; Create GSO and enter description of GSO resource</td>
</tr>
<tr>
<td>rsrc delete resource_name</td>
<td>GSO Resource &gt; List GSO &gt; select GSO resources &gt; Delete</td>
</tr>
<tr>
<td>rsrc list</td>
<td>GSO Resource &gt; List GSO</td>
</tr>
<tr>
<td>rsrccred create resource_name rsrccred resource_userid rsrccpwd resource_pwd rsrctype {web|group} user user_name</td>
<td>User &gt; Search Users &gt; Search &gt; click user name &gt; click GSO Credentials tab &gt; click Create</td>
</tr>
<tr>
<td>rsrccred create resource_group_name rsrccred resource_userid rsrccpwd resource_pwd rsrctype {web|group} user user_name</td>
<td>User &gt; Search Groups &gt; Search &gt; click user name &gt; click GSO Credentials tab &gt; click Create</td>
</tr>
<tr>
<td>rsrccred delete resource_name rsrctype {web|group} user user_name</td>
<td>User &gt; Search Users &gt; Search &gt; click user name &gt; click GSO Credentials tab &gt; select GSO Credentials &gt; Delete</td>
</tr>
<tr>
<td>rsrccred delete resource_group_name rsrctype {web|group} user user_name</td>
<td>User &gt; Search Groups &gt; Search &gt; click user name &gt; click GSO Credentials tab &gt; select GSO Credentials &gt; Delete</td>
</tr>
<tr>
<td>rsrccred list user user_name</td>
<td>User &gt; Search Users &gt; Search &gt; click user name &gt; click GSO Credentials tab</td>
</tr>
<tr>
<td>rsrccred modify resource_name rsrctype {web|group} [-rsrccred resource_userid] [-rsrccpwd resource_pwd] user user_name</td>
<td>User &gt; Search Users &gt; Search &gt; click user name &gt; click GSO Credentials tab &gt; click Create</td>
</tr>
</tbody>
</table>
Table 12. Mapping between administration CLI and Web Portal Manager (continued)

<table>
<thead>
<tr>
<th>pdadmin CLI</th>
<th>Web Portal Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>rsrccred modify resource_group_name rsrctype {web</td>
<td>group} [-rsruser resource_userid] [-rsrwp resource_pwd] user user_name</td>
</tr>
<tr>
<td>rsrccred show resource_name rsrctype {web</td>
<td>group} user user_name</td>
</tr>
<tr>
<td>rsrccred show resource_group_name rsrctype {web</td>
<td>group} user user_name</td>
</tr>
<tr>
<td>rsrgroup create resource_group_name</td>
<td>GSO Resource → Create GSO Group</td>
</tr>
<tr>
<td>rsrgroup create resource_group_name -desc description</td>
<td>GSO Resource → Create GSO Group and enter the description</td>
</tr>
<tr>
<td>rsrgroup delete resource_group_name</td>
<td>GSO Resource → List GSO Groups → select GSO resource groups → Delete</td>
</tr>
<tr>
<td>rsrgroup list</td>
<td>GSO Resource → List GSO Groups</td>
</tr>
<tr>
<td>rsrgroup modify resource_group_name add rsrccname resource_name</td>
<td>GSO Resource → List GSO Groups → click GSO resource group → Add</td>
</tr>
<tr>
<td>rsrgroup modify resource_group_name remove rsrccname resource_name</td>
<td>GSO Resource → List GSO Groups → click GSO resource group → select members → Remove</td>
</tr>
<tr>
<td>rsrgroup show resource_group_name</td>
<td>GSO Resource → List GSO Groups → click GSO resource group</td>
</tr>
<tr>
<td>server list</td>
<td>Not supported</td>
</tr>
<tr>
<td>server listtasks server_name</td>
<td>Not supported</td>
</tr>
<tr>
<td>server replicate server_name</td>
<td>Not supported</td>
</tr>
<tr>
<td>server show server_name</td>
<td>Not supported</td>
</tr>
<tr>
<td>server task server_name [help</td>
<td>stats</td>
</tr>
<tr>
<td>server task (WebSEAL) server_name server_task</td>
<td>Not supported</td>
</tr>
<tr>
<td>user create [-gsouser] [-no-password-policy] user_name dn cn sn password</td>
<td>User → Create User</td>
</tr>
<tr>
<td>[([group1 group2 .... ])]</td>
<td></td>
</tr>
<tr>
<td>user delete [-registry] user_name</td>
<td>User → Search Users → enter pattern and maximum results → Search → select user names → Delete</td>
</tr>
<tr>
<td>user import [-gsouser] user_name dn [group_name]</td>
<td>User → Import User</td>
</tr>
<tr>
<td>user list pattern max_return</td>
<td>User → Search Users → enter pattern and maximum results → Search</td>
</tr>
<tr>
<td>user list-dn pattern max_return</td>
<td>Not supported</td>
</tr>
<tr>
<td>user modify user_name account-valid {yes</td>
<td>no}</td>
</tr>
</tbody>
</table>
Table 12. Mapping between administration CLI and Web Portal Manager (continued)

<table>
<thead>
<tr>
<th>pdadmin CLI</th>
<th>Web Portal Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>user modify user_name password password</code></td>
<td>User → Search Users → enter pattern and maximum results → Search → click user name → General tab</td>
</tr>
<tr>
<td>`user modify user_name password-valid {yes</td>
<td>no}`</td>
</tr>
<tr>
<td><code>user show user_name</code></td>
<td>User → Search Users → enter pattern and maximum results → Search → click user name</td>
</tr>
<tr>
<td><code>user show-dn dn</code></td>
<td>Not supported</td>
</tr>
<tr>
<td><code>user show-groups user_name</code></td>
<td>User → Search Users → enter pattern and maximum results → Search → click user name → Groups tab</td>
</tr>
</tbody>
</table>
Appendix D. Managing user registries

This chapter contains a subset of user registry tasks that are specific to the installation of Tivoli Access Manager. For common administrative tasks for your particular registry (tasks that are not Tivoli Access Manager-specific), refer to the documentation that came with your user registry product.

This chapter contains the following sections:

- “LDAP-specific tasks”
- “Active Directory-specific tasks” on page 351
- “Novell-specific tasks” on page 353

LDAP-specific tasks

LDAP is a protocol that runs over TCP/IP. The LDAP protocol standard includes low-level network protocol definitions plus data representation and handling functionality. A directory that is accessible through LDAP is commonly referred to as an LDAP directory.

This section contains the following topics:

- “LDAP failover configuration”
- “Using valid characters for LDAP user and group names” on page 342
- “Applying IBM Tivoli Access Manager ACLs to new LDAP suffixes” on page 343

LDAP failover configuration

The Lightweight Directory Access Protocol (LDAP) defines a standard method for accessing and updating information in a directory. Directories are usually accessed using the client/server model of communication. Any server that implements the LDAP protocol is an LDAP directory server.

The LDAP distributed architecture supports scalable directory services with server replication capabilities. Server replication improves the availability of a directory service. IBM Directory replication is based on a master-subordinate model. Sun ONE Directory Server replication is based on a supplier/consumer model. Tivoli Access Manager still treats this as a master-subordinate relationship.

The combination of a master server and multiple replicated servers helps ensure that directory data is always available when needed. If any server fails, the directory service continues to be available from another replicated server. Tivoli Access Manager supports this replication capability.

The master-subordinate replication model

Replication involves two types of directories: master and replica. LDAP refers to the master as master server and to the replica as replica server. All updates are made on the master server and these updates are subsequently propagated to the replica servers. Each replica server database contains an exact copy of the master server’s directory data.

Changes to the directory can be made only to the master server, which is always used for write operations to the directory. Either the master or the replicas can be
used for read operations. When the original master server is out of service for an extended period of time, a replica server can be promoted as a master server to allow write operations to the directory.

**Tivoli Access Manager failover capability for LDAP servers**

Tivoli Access Manager connects to the LDAP master server when it starts up. If the LDAP master server is down for any reason, the Tivoli Access Manager server must be able to connect to an available LDAP replica server for any read operations.

Many operations, especially those from regular users, are read operations. These include operations such as user authentication and signon to backend junctioned Web servers. After proper configuration, Tivoli Access Manager performs failover to a replica server when it cannot connect to the master server.

You can find the configuration parameters for LDAP failover in the [ldap] stanza of the ldap.conf configuration file:

UNIX: `/opt/PolicyDirector/etc/ldap.conf`
Windows: `install_path\etc\ldap.conf`

**Master server configuration**

IBM Directory supports the existence of a single read-write master LDAP server. Sun ONE Directory Server supports multiple read-write LDAP servers. Tivoli Access Manager treats the Sun ONE supplier server as the master server for configuration purposes.

The active configuration lines in the ldap.conf file represent the parameters and values for this master LDAP server. You determine these values during Tivoli Access Manager configuration. For example:

```
[ldap]
enabled = yes
host = outback
port = 389
ssl-port = 636
max-search-size = 2048
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enabled</code></td>
<td>Tivoli Access Manager uses an LDAP user registry. Values are yes and no.</td>
</tr>
<tr>
<td><code>host</code></td>
<td>The network name of the machine where the LDAP master server is located.</td>
</tr>
<tr>
<td><code>port</code></td>
<td>The TCP listening port of the LDAP master server.</td>
</tr>
<tr>
<td><code>ssl-port</code></td>
<td>The SSL listening port of the LDAP master server.</td>
</tr>
<tr>
<td><code>max-search-size</code></td>
<td>The Tivoli Access Manager limit for an LDAP client search of database items - such as a request for the Web Portal Manager to list users from the LDAP database.</td>
</tr>
</tbody>
</table>

If you make a change to the LDAP database, such as adding a new user account through the Web Portal Manager, Tivoli Access Manager always uses the read-write (master) LDAP server.
Replica server configuration
IBM Directory supports the existence of one or more read-only replica LDAP servers. Sun ONE Directory Server supports the existence of one or more read-only replica LDAP servers referred to as consumers.

You must add lines to the [ldap] stanza that identifies any replica servers available to Tivoli Access Manager. Use the following syntax for each replica:

```
replica = ldap_server, port, type, preference
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldap-server</td>
<td>The network name of the LDAP replica server.</td>
</tr>
<tr>
<td>port</td>
<td>The port this server listens on. Generally, use 389 or 636.</td>
</tr>
<tr>
<td>type</td>
<td>The functionality of the replica server - either read-only or read-write.</td>
</tr>
<tr>
<td></td>
<td>Normally, use read-only. A read-write type would represent a master server.</td>
</tr>
<tr>
<td>preference</td>
<td>A number from 1 - 10. The server with the highest preference value is chosen for LDAP connections. See &quot;Setting preference values for replica LDAP servers.&quot;</td>
</tr>
</tbody>
</table>

Example:

```
replica = replica1.ldap.tivoli.com,389,readonly,5
replica = replica2.ldap.tivoli.com,389,readonly,5
```

Changes to the ldap.conf file do not take effect until you restart Tivoli Access Manager.

Setting preference values for replica LDAP servers
Each replica LDAP server must have a preference value (1-10) that determines its priority for selection as:

- The primary read-only access server, or
- A backup read-only server during a failover

The higher the number, the higher the priority. If the primary read-only server fails for any reason, the server with the next highest preference value is used. If two or more servers have the same preference value, a least-busy load balancing algorithm determines which one is selected.

Remember that the master LDAP server can function as both a read-only and a read-write server. For read-only access, the master server has a hard-coded default preference setting of 5. This preference setting allows you to set replica servers at values higher or lower than the master to obtain the required performance. For example, with appropriate preference settings, you could prevent the master server from handling everyday read operations.

You can set hierarchical preference values to allow access to a single LDAP server (with failover to the other servers), or set equal preferences for all servers and allow load balancing to dictate server selection.

The following table illustrates some possible preference scenarios. “M” refers to the master (read-only/read-write) LDAP server; “R1, R2, R3” refer to the replica (read-only) LDAP servers.
<table>
<thead>
<tr>
<th>M</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>Failover preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>All servers have the same preference values. Load balancing determines which server is selected for each access operation.</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>The three replica servers have the same preference value. This value is higher than the master server value. Load balancing determines server selection among the three replicas. The master is used only if all three replica servers become unavailable.</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>Server 3 (with the highest preference value) becomes the primary server. If server 3 fails, server 2 becomes the primary server because it has the next highest preference value.</td>
</tr>
</tbody>
</table>

Preference values affect only read-only access to the LDAP database. Tivoli Access Manager always uses the master (read-write) server when you need to make a change to the LDAP database.

Also note that some Tivoli Access Manager daemons (such as the policy server) override the preference settings in their configuration files to indicate that the read-write server is preferred. This override occurs because those daemons usually make update operations that should go to the master LDAP server.

**Server polling**
If an LDAP server does fail, Tivoli Access Manager continuously polls the server to check for its return to active duty. The poll time is 10 seconds.

### Using valid characters for LDAP user and group names
When using LDAP as the user registry, the set of valid characters allowed within a user or group name is determined by the following Internet Engineering Task Force (IETF) Request for Comments (RFC):

- 2254 "The String Representation of LDAP Search Filters"

The specific LDAP server can also dictate the validity of these characters.

In general, you can use special characters within a Distinguished Name. However, certain special characters require an additional escape character. The following special characters must be escaped when used in a Distinguished Name:

- + (plus)
- \ (backslash)
- ; (semicolon)
- , (comma)

For example, to create a user containing a semicolon using the `pdadmin` utility:

```bash
pdadmin> user create "user;one" "cn=user\;one,o=tivoli,c=us"
"user;one" "user;one" password1
```

If you use special characters when using `pdadmin` from a command line, enclose each argument of the user or group command with double quotation marks. The double quotation marks allow the argument to be entered without being subject to interpretation by the operating system shell command processor.
Due to the variability of special character handling in general, avoid using special characters.

**Applying IBM Tivoli Access Manager ACLs to new LDAP suffixes**

The LDAP naming model is usually maintained in a hierarchical namespace referred to as the Directory Information Tree (DIT). Many LDAP server products, such as the IBM Tivoli Directory Server product included with Tivoli Access Manager, and the Sun ONE Directory Server, maintain the data of the DIT in a hierarchical namespace often represented as a tree structure. The top of the tree is referred to as a naming context, also sometimes simply called a suffix because it represents the ending portion of a distinguished name (DN) of an entry in LDAP. For example, a suffix called c=us might be created to represent data within an organization for that country. A particular entry within this suffix would have a DN similar to cn=Joe Williams,ou=austin,o=ibm,c=us. The set of suffixes, which the LDAP server maintains, is configurable using the administration tools associated with the LDAP server.

When the Tivoli Access Manager policy server is configured, it attempts to apply appropriate access control, in the form of Access Control Lists (ACLs) to every LDAP suffix that exists at that time in the LDAP server. This access control gives appropriate permission to allow Tivoli Access Manager to create and manage user and group information within these LDAP suffixes.

If an LDAP administrator adds an LDAP suffix after Access Manager has been configured and wants Access Manager to be able to manage users and groups within this new suffix, the appropriate ACLs should be applied to the new suffix manually.

To apply the appropriate access controls to the newly created LDAP suffix, use the appropriate LDAP administration interface to apply the following ACLs to every new suffix:

<table>
<thead>
<tr>
<th>LDAP Group</th>
<th>Access Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>cn=SecurityGroup,secAuthority=Default</td>
<td>• Full access</td>
</tr>
<tr>
<td>cn=ivacl-servers,cn=SecurityGroups,secAuthority=Default</td>
<td>• read</td>
</tr>
<tr>
<td></td>
<td>• search</td>
</tr>
<tr>
<td></td>
<td>• compare</td>
</tr>
<tr>
<td>cn=remote-acl-users,cn=SecurityGroups,secAuthority=Default</td>
<td>• read</td>
</tr>
<tr>
<td></td>
<td>• search</td>
</tr>
<tr>
<td></td>
<td>• compare</td>
</tr>
</tbody>
</table>

In addition, if the Access Manager administrator has created multiple domains, that is, more than one administrative domain other than the initial Management Domain, which gets created when the Policy Server is first configured, the following additional ACLs should be applied to the new suffix for each domain.
<table>
<thead>
<tr>
<th>LDAP Group</th>
<th>Access Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>cn=SecurityGroup,secAuthority=domain_name,cn=Subdomains,secAuthority=Default</td>
<td>• Full control</td>
</tr>
<tr>
<td>cn=ivacl-servers,cn=SecurityGroups,secAuthority=domain_name,cn=Subdomains,secAuthority=Default</td>
<td>• read</td>
</tr>
<tr>
<td></td>
<td>• search</td>
</tr>
<tr>
<td></td>
<td>• compare</td>
</tr>
<tr>
<td>cn=remote-acl-users,cn=SecurityGroups,secAuthority=domain_name,cn=Subdomains,secAuthority=Default</td>
<td>• read</td>
</tr>
<tr>
<td></td>
<td>• search</td>
</tr>
<tr>
<td></td>
<td>• compare</td>
</tr>
</tbody>
</table>

Where *domain_name* is the name of the additional administrative domain. For a list of domains, use the `pdadmin domain list` command.

The following example steps can be used for either the IBM Directory Server or for the Sun ONE Directory Server, depending on the LDAP server type being used. Note that the procedures being described assume the newly created suffix is called c=fr. You should substitute the actual newly created suffix for this value in the following descriptions.

**Procedures for the IBM Tivoli Directory Server**

The following steps describe how to apply the appropriate Tivoli Access Manager access controls to the newly created suffix for the IBM Tivoli Directory Server. These steps use the IBM Directory Server Web Administration Tool, which is included with IBM Tivoli Directory Server version 5.2, and assume that the tool has been properly installed and configured into the WebSphere application server.

1. Access the login page using a supported Web Browser. The default login page URL is

   `http://server_name:9080/IDSWebApp/IDSjsp/Login.jsp`

   where *server_name* is the host name of the application server where the IBM Directory Server Web Administration Tool has been installed.

2. Use the list to select the LDAP server host name to be administered and go to step 7 on page 345.

   If you have not already added the LDAP host to the list to be administered, you will need to log in as the Console Admin and add the LDAP server to the list of console servers. To do this, continue to step 3.

3. Log in as the Console Admin. The default Console Admin identity is `superadmin` and the default password is `secret`.

4. In the navigation area on the left, click **Console administration** and **Manage console servers**. This action presents a list of LDAP servers that are currently configured for administration. To add another LDAP server, click **Add** and then type the host name and port number information for the LDAP server to be administered. When complete, click **OK**.

5. After you have added the LDAP servers to be administered, click **Close** to complete the **Manage console servers** action. Then, click **Logout** from the navigation area.

6. Re-access the login page using the same URL as in step 1 and select the LDAP server that you added from the list.
7. After you have selected the LDAP server from the list, type the LDAP server administrator Username (cn=root) and password on the Login window. Click Login.
8. In the navigation area on the left, click Directory management and Manage entries.
9. If you do not see the newly added suffix in the Manage entries window shown on the right, it indicates that no entry for the newly added suffix has yet been added. Before access control can be applied to the suffix, an entry must first be created. If you do see the newly added suffix, go to step 13. Otherwise, to add an entry for a newly created suffix, click Add. The Add an entry window is displayed.
10. Select the appropriate structural object class for the newly added suffix. For the c=fr suffix, the appropriate object class is country. Click Next to proceed to the next window. The Select auxiliary object classes window is displayed.
11. This window allows you to add additional object classes appropriate for the entry type. In this example, no other object classes are needed so just click Next.
12. The following window allows you to name the new entry and enter attribute values for the structural object class chosen. For this example, enter the Relative DN as c=fr and leave the Parent DN blank. The only required attribute for this example is c for country. Fill in the value fr for this example and then click Finish. This will take you back to the Manage entries window and you should now see the newly added suffix in the list of top-level entries.
13. Select the suffix from the list by clicking on it in the Select column, and then click Edit ACL. The Edit ACL window is displayed and shows the current ACLs on the suffix entry. Click Non-filtered ACLs.
14. Be sure the Propagate ACLs option is selected. Enter the following group name in the DN (distinguished name) field: cn=SecurityGroup,secAuthority=Default. Set the Type to group and click Add. The Add access rights window is displayed.
15. Set the Add child and Delete entry right to grant. Set all Security classes (normal, sensitive, critical, system and restricted) to grant for all actions (read, write, search, compare). Then click OK. The Edit ACL window is re-displayed.
16. Enter the following group name in the DN (distinguished name) field: cn=ivacl-servers,cn=SecurityGroups,secAuthority=Default. Set the Type to group and click Add. The Add access rights window is displayed.
17. Set the normal and system Security classes to grant for the read, search and compare actions. Leave the Add child, Delete entry, and all other Security classes blank. Then click OK. The Edit ACL window is re-displayed.
18. Enter the following group name in the DN (distinguished name) field: cn=remote-acl-users,cn=SecurityGroups,secAuthority=Default. Set the Type to group and click Add. The Add access rights window is displayed.
19. Set the normal and system Security classes to grant for the read, search and compare actions. Leave the Add child, Delete entry, and all other Security classes blank. Then click OK. The Edit ACL window is re-displayed.
20. If you have no further domains, this completes the access control and you can skip to step 27 on page 346. If you have domains and need to add the domain ACLs, continue with step 21.
21. Enter the following group name in the DN (distinguished name) field: cn=SecurityGroup,secAuthority=domain_name,cn=Subdomains,
secAuthority=Default. Where domain_name is the domain name being protected. Set the Type to group and click Add. The Add access rights window is displayed.

22. Set the Add child and Delete entry right to grant. Set all Security classes (normal, sensitive, critical, system and restricted) to grant for all actions (read, write, search, compare). Then click OK. The Edit ACL window is re-displayed.

23. Enter the following group name in the DN (distinguished name) field:
cn=ivacl-servers,cn=SecurityGroups,secAuthority=domain_name,cn=Subdomains, secAuthority=Default. Set the Type to group and click Add. The Add access rights window is displayed.

24. Set the normal and system Security classes to grant for the read, search and compare actions. Leave the Add child, Delete entry, and all other Security classes blank. Then click OK. The Edit ACL window is re-displayed.

25. Enter the following group name in the DN (distinguished name) field:
cn=remote-acl-users,cn=SecurityGroups,secAuthority=domain_name,cn=Subdomains, secAuthority=Default. Set the Type to group and click Add. The Add access rights window is displayed.

26. Set the normal and system Security classes to grant for the read, search and compare actions. Leave the Add child, Delete entry, and all other Security classes blank. Then click OK. The Edit ACL window is re-displayed.

If you have further domains, repeat steps 21 on page 345 to 26 for each domain. When complete, continue with step 27

27. This completes the addition of the access control for the suffix. Click OK. The Manage entries window is re-displayed. Click Close. The LDAP server does not need to be restarted for the changes to take effect.

28. If you are finished with the IBM Directory Server Web Administration Tool, click Logout.

Procedures for the Sun ONE Directory Server
The following steps describe how to apply the appropriate Tivoli Access Manager access controls to the newly created suffix for the Sun ONE Directory Server. These steps use the Sun ONE Server Console Version 5.2.

1. Start the Sun ONE Server Console using one of the following commands:
   • On UNIX systems, enter the following from the Sun ONE Directory Server install directory:
     # ./startconsole
   • On Solaris, when not using the Solaris packaged version, change to the server root directory, and then enter:
     startconsole arguments
     Type -h to display a usage message explaining command line arguments.
   • On Windows systems, click: Start → Programs → Sun ONE Server Products → Sun ONE Server Console Version 5.2.

2. Log in to the Sun ONE Server Console. Type the user ID for the LDAP administrator, which is usually cn=Directory Manager. Type the password and the Administration URL. Click OK.

3. Select the Sun ONE Domain to be used by Tivoli Access Manager.

4. Expand the server name and Server Group.

5. Select the entry labeled Directory Server. Configuration information about the Sun ONE Directory server is displayed.
6. Click **Open**. The Sun ONE Directory server is accessed.
7. Click the **Directory** tab. If the newly created suffix is displayed on the left pane, skip to step [9]
   If the newly created suffix does not appear in the left pane, you must create an entry for the new suffix before applying access controls to the suffix. Follow these steps to create the entry:
   a. Highlight the name of the server at the top of the directory tree. Click Object → New Root Object. A list of root suffixes is displayed.
   b. Select c=fr from the list of root suffixes. The New Object selection window is displayed.
   c. In the New Object selection window, scroll down and select **Country** as the new object entry type.
   d. Click **OK**. The Property Editor window is displayed.
   e. Fill in the **Country** field as fr and click **OK**.

   **Note:** These instructions assume an example suffix. Create the entry type and name that corresponds to your actual suffix.
   f. Click **View → Refresh**. The new suffix entry appears in the left navigation tree.
8. Highlight the c=fr entry in the left pane. Click Object → **Set Access Permissions**. The Manage Access Control for c=fr window is displayed.
9. Click **New** to display the Edit ACI for c=fr window.
10. Specify the Access Control Instructions (ACI) name as SECURITY GROUP – ALLOW ALL.
11. Highlight the **All Users** name and click **Remove**.
12. Click **Edit Manually**. The Edit ACI for c=fr window is displayed
13. Replace the default ACI text with the following:
   ```
   (target="ldap:///c=fr")(targetattr="*")
   (version 3.0; acl "SECURITY GROUP – ALLOW ALL"
   allow (all)
   groupdn = "ldap:///cn=SecurityGroup,secAuthority=Default";)
   ```
   Click **Check Syntax** to ensure that you have entered the text correctly. Correct any errors until the syntax passes the check.
14. Click **OK**. The Manage Access Control for c=fr window is displayed.
15. Click **New**. Specify the ACI name as PD Servers GROUP – ALLOW READ.
16. Highlight the **All Users** name and click **Remove**.
17. Click **Edit Manually**. The edit ACI for c=fr window is displayed.
18. Replace the default ACI text with the following:
   ```
   (target="ldap:///c=fr")(targetattr="*")
   (version 3.0; acl "PD Servers GROUP – ALLOW READ"
   allow (read, search, compare)
   groupdn = "ldap:///cn=ivacld-servers,cn=SecurityGroups,secAuthority=Default";)
   ```
   Click **Check Syntax** to ensure that you have entered the text correctly. Correct any errors until the syntax passes the check.
19. Click **OK**. The Manage Access Control for c=fr window is displayed.
20. Click **New**. Specify the ACI name as PD Remote ACL Users GROUP – ALLOW READ.
21. Highlight the **All Users** name and click **Remove**.
22. Click **Edit Manually**. The Edit ACI for c=fr window is displayed.
23. Replace the default ACI text with the following:

```
(target="ldap://c=fr")

(version 3.0; acl "PD Remote ACL Users GROUP – ALLOW READ"
allow (read, search, compare)
groupdn = "ldap://cn=remote-acl-users,cn=SecurityGroups,
secAuthority=Default";)
```

Click **Check Syntax** to ensure that you have entered the text correctly. Correct any errors until the syntax passes the check.

24. Click **OK**. The Manage Access Control for c=fr window is displayed.

25. Click **New**. Specify the ACI name as PD Deny-Others.

26. Highlight the **All Users** name and click **Remove**.

27. Click **Edit Manually**. The Edit ACI for c=fr window is displayed.

28. Replace the default ACI text with the following:

```
(target="ldap://c=fr")

(version 3.0; acl "PD Deny-Others"
deny(all)
groupdn = "ldap://cn=remote-acl-users,cn=SecurityGroups,
secAuthority=Default"
ldap:///cn=ivacld-servers,cn=SecurityGroups,secAuthority=Default||
ldap:///cn=remote-acl-users,cn=SecurityGroups,secAuthority=Default||
ldap:///cn=ivacld-servers,cn=SecurityGroups,secAuthority=Default;)
```

Click **Check Syntax** to ensure that you have entered the text correctly. Correct any errors until the syntax passes the check.

29. Click **OK**. The Manage Access Control for c=fr window is displayed.

30. If you have no further domains, this completes the access control. You can skip to step 52 on page 349.

   If you have domains and need to add the domain ACLs, continue with step 31.

31. Click **New**. Specify the ACI name as SECURITY GROUP – ALLOW ALL.

32. Highlight the **All Users** name and click **Remove**.

33. Click **Edit Manually**. The Edit ACI for c=fr window is displayed.

34. Replace the default ACI text with the following:

```
(target="ldap://c=fr")

(version 3.0; acl "SECURITY GROUP – ALLOW ALL"
allow (all)
groupdn = "ldap://cn=SecurityGroup,secAuthority=domain_name,
    cn=Subdomains,secAuthority=Default";)
```

where *domain_name* is the name of the domain being protected.

Click **Check Syntax** to ensure that you have entered the text correctly. Correct any errors until the syntax passes the check.

35. Click **OK**. The Manage Access Control for c=fr window is displayed.

36. Click **New**. Specify the ACI name as PD Servers GROUP – ALLOW READ.

37. Highlight the **All Users** name and click **Remove**.

38. Click **Edit Manually**. The Edit ACI for c=fr window is displayed.

39. Replace the default ACI text with the following:

```
(target="ldap://c=fr")

(version 3.0; acl "PD Servers GROUP – ALLOW READ"
allow (read, search, compare)
groupdn = "ldap://cn=ivacld-servers,cn=SecurityGroups,
    secAuthority=domain_name,cn=Subdomains,secAuthority=Default";)
```

where *domain_name* is the name of the domain being protected.
Click Check Syntax to ensure that you have entered the text correctly. Correct any errors until the syntax passes the check.

40. Click OK. The Manage Access Control for c=fr window is displayed.

41. Click New. Specify the ACI name as PD Remote ACL Users GROUP - ALLOW READ.

42. Highlight the All Users name and click Remove.

43. Click Edit Manually. The Edit ACI for c=fr window is displayed.

44. Replace the default ACI text with the following:

   
   (target="ldap:///c=fr")
   
   (version 3.0; acl "PD Remote ACL Users GROUP - ALLOW READ"
   
   allow (read, search, compare)
   
   groupdn = "ldap:///cn=remote-acl-users,cn=SecurityGroups,secAuthority=default"
   
   )

   
   where domain_name is the name of the domain being protected.

   
   Click Check Syntax to ensure that you have entered the text correctly. Correct any errors until the syntax passes the check.

45. Click OK. The Manage Access Control for c=fr window is displayed.

46. Click New. Specify the ACI name as PD Deny-Others.

47. Highlight the All Users name and click Remove.

48. Click Edit Manually. The Edit ACI for c=fr window is displayed.

49. Replace the default ACI text with the following:

   
   (targetfilter="(secAuthority=domain_name)"
   
   (version 3.0; acl "PD Deny-Others"
   
   deny(all)
   
   groupdn != "ldap:///cn=SecurityGroup,secAuthority=default"
   
   "ldap:///cn=SecurityGroup,secAuthority=domain_name, cn=Subdomains,secAuthority=Default"
   
   "ldap:///cn=remote-acl-users,cn=SecurityGroups,secAuthority=domain_name, cn=Subdomains,secAuthority=Default"
   
   "ldap:///cn=ivacl-servers,cn=SecurityGroups,secAuthority=domain_name, cn=Subdomains,secAuthority=Default"
   
   )

50. Click OK. The Manage Access Control for c=fr window is displayed.

51. If there are further domains, repeat steps 31 on page 348 to 50 For each domain. When complete, continue with step 52.

52. Click OK to close the Manage Access Control for c=fr window.

53. Click Console → Exit to exit the console.

Procedure for the IBM z/OS and OS/390 Security Servers

After the Tivoli Access Manager policy server has been configured, this procedure describes how to apply the appropriate Tivoli Access Manager access controls to the newly created suffix. These steps are specifically for the IBM z/OS Security Server LDAP Server Version 1, Release 2 or higher, and the IBM Security Server for OS/390 Version 2, Release 10. Hereafter, both servers are referred to collectively as the IBM z/OS and OS/390 Security Servers.

To add a suffix for the IBM z/OS or OS/390 Security Server:

1. Add the new suffix to the security server slapd.conf file. See the z/OS LDAP Server Administration and Use Guide for details on how to update the IBM z/OS or OS/390 Security Server configuration file.

2. Restart the IBM z/OS or OS/390 Security Server.

3. To add an entry to the newly created suffix, do the following:
a. Create an LDIF file. This example assumes the newly created suffix is
   o=neworg,c=us:
   dn: o=neworg,c=us
   objectClass: organization
   objectClass: top
   o: neworg

b. Use the appropriate LDIF file as input to the `ldapadd` command:
   
   ```
   ldapadd -D ldap_admin -w ldap_pwd -v -f ldif_filename
   ```

4. To apply the appropriate Tivoli Access Manager access controls to the newly
   created suffix (suffix), do the following:
   
   a. If no additional Access Manager domains have been created other than the
      initial management domain, create this LDIF file:

      ```
      suffix
      aclpropagate=TRUE
      aclentry=group:cn=ivacld-servers,cn=securitygroups,secauthority=default: normal:csr
      aclentry=group:cn=remote-acl-users,cn=securitygroups,secauthority=default: normal:csr
      suffix
      ownerpropagate=TRUE
      entryOwner=group:cn=SecurityGroup,secAuthority=Default
      entryOwner=access-id:LDAP_admin_dn
      ```

      The backward slash ( `\` ) at the end of a line indicates that this line
      combines with the next line, without any spaces.

   b. Apply the updates in the LDIF file by using it as input to the `ldapmodify` 
      command:

      ```
      ldapmodify -h hostname -D LDAP_admin_dn -w admin_pwd -v -f ldif_filename
      ```

   c. However, if a domain has been created in addition to the initial
      management domain and if a new suffix is created, ACLs will have to be
      applied for each added domain also. Add ACLs to each added domain by
      creating an LDIF file similar to:

      ```
      suffix
      aclentry=group:cn=ivacld-servers,cn=securitygroups,secauthority=default:\
      added_domain,cn=subdomains,secauthority=default:normal:csr
      aclentry=group:cn=remote-acl-users,cn=securitygroups,secauthority=default:\
      added_domain,cn=subdomains,secauthority=default:normal:csr
      aclentry=group:cn=securitygroup,secauthority=added_domain,\ cn=subdomains,secauthority=default:object:ad:normal:\ rwsc:sensitive:\
      rwsc:critical:rwsc:restricted:rwsc
      ```

   d. Apply the updates in the LDIF file by using it as input to the `ldapmodify` 
      command:

      ```
      ldapmodify -h hostname -D LDAP_admin_dn -w LDAP_admin_pwd -v -f ldif_filename
      ```

   **Note:** If aclpropagate=TRUE is set by default for the added suffix, the `ldapmodify`
   command will return this error:

   ```
   ldapmodify: additional info: R004086 Entry o=neworg,c=us already contains
   attribute aclpropagate, value=TRUE
   ```

   In this case, remove aclpropagate=TRUE from the LDIF file, and rerun the
   `ldapmodify` command.
Active Directory-specific tasks

Microsoft Active Directory is an infrastructure supported by Windows 2000 that includes a network management of directory objects, and has the capability to communicate with other directory services.

This section contains the following topics:

- “Setting up Microsoft Windows 2000 Domain Name System for Active Directory”
- “Updating the Tivoli Access Manager schema” on page 352
- “Adding a Tivoli Access Manager user to the Active Directory system group” on page 352

Setting up Microsoft Windows 2000 Domain Name System for Active Directory

Active Directory uses the Domain Name System (DNS) as a domain controller location mechanism. DNS enables computers to find the IP addresses of the domain controllers.

For multi-domain mode, at least two domains are required from these types of domains:

- A primary domain
- A child domain of the primary domain
- A domain tree in the forest

For failover, at least two primary domain controllers are needed.

You can set up the DNS server before configuring the domain controllers or when you configure the primary Active Directory domain controller. There are two ways to setup DNS for Active Directory:

1. Configure DNS on the forest root
2. Use a separate DNS server

If configuring DNS on the forest root, DNS is configured automatically on that host if this is the first domain controller configured. This domain controller and its replicas serve as the DNS servers.

The DNS server is not necessary on the host that is the domain controller in the forest. You can use any DNS server. If you are not using a Windows 2000-based DNS server, contact your DNS administrator or a DNS server vendor to find out whether your server supports the required standards. If the server does not support the required standards or the zone cannot be configured to allow dynamic updates, you need to modify the existing DNS infrastructure.

Adding a new domain name to a DNS

To add a new domain name to a DNS, do the following:

1. Click Start⇒Programs⇒Administrator Tools⇒DNS to open the DNS.
2. Expand the host name and Forward Lookup Zones.
3. Create a new zone (new root domain) or child domain.
4. If using a separate DNS, open the domain properties and change the Allow dynamic updates field to Yes.
Updating the Tivoli Access Manager schema

In order to perform all Tivoli Access Manager operations, you need to add a Tivoli Access Manager schema on Active Directory. The Tivoli Access Manager schema adds to the schema master, which is a root domain controller in the forest. The Tivoli Access Manager schema automatically updates to the schema master during Tivoli Access Manager configuration. You need to manually update the Tivoli Access Manager schema only when Tivoli Access Manager is configured to a single Active Directory domain (not the root domain).

Note: Before updating the Tivoli Access Manager schema, verify that it is not already on the schema master. The Tivoli Access Manager schema needs to be updated only once in the forest.

To verify if the Tivoli Access Manager schema is updated on your system, do the following:
1. In your domain controller, go to Start → Programs → Administrative Tools → Active Directory Users and Computers. The Active Directory Users and Computers dialog appears.
2. In this dialog, expand the domain that contains the Users folder.
3. Right click on the Users folder. A menu appears.
4. Click New in the menu. Another menu appears.
5. If a list of Tivoli Access Manager classes for Active Directory appear in the menu in the URAF-xxx form, (for example, URAF-container), then the Tivoli Access Manager schema is already on the schema master. You do not need to update the Tivoli Access Manager schema.

To manually update the Tivoli Access Manager schema, do the following:
1. Install Tivoli Access Manager runtime on the root domain controller.
2. Run the \aminstall_dir\sbin\adschema_update -u AMConfID -p AMConfPWD command
   where:
   • \aminstall_dir is the directory that installs Tivoli Access Manager
   • AMConfID is the Tivoli Access Manager configuration login ID
   • AMConfPWD is the Tivoli Access Manager configuration login password
3. After you verify that the Tivoli Access Manager schema has been added to the schema master, you can uninstall Tivoli Access Manager runtime from the root domain.

Note: The Tivoli Access Manager schema propagation takes approximately five minutes from the schema master to add to the non-root domain controller.

Adding a Tivoli Access Manager user to the Active Directory system group

In order to have sufficient access to modify user and group attributes, a Tivoli Access Manager user must be added to the appropriate Active Directory system group. To add a user to an Active Directory system group on a system where Active Directory is configured as a Tivoli Access Manager user registry, and do the following:
1. Log in as Administrator.
2. Go to Start → Programs → Administrative Tools.
3. Click **Active Directory Users and Computers** from the menu. The Active Directory Users and Computers window appears.

4. On the left navigation panel, go to **Tivoli PD Domains → default → system → users**, where the users container of the Tivoli Access Manager user registry container is located.

5. From the list of users displayed, select the Tivoli Access Manager user that you want to add to the Active Directory system group.

6. Right-click on the Tivoli Access Manager user, and click **Properties**. The Properties window for the selected Tivoli Access Manager user is displayed.

7. Click the **Member Of** tab.

8. Click **Add**. The Select Groups window appears.

9. Select the appropriate group that you want the Tivoli Access Manager user to become a member of, and click **Add**.

10. Do one of the following:
   - If the purpose is to modify user or group attributes for Active Directory single domain, select the **Domain Admins** group.
   - If Tivoli Access Manager is configured using Active Directory multiple domain, select the **Enterprise Admins** group.

11. For each user you want to add to multiple groups, repeat the add-user-to-group process.

12. Click **OK** to close all opened windows.

---

**Novell-specific tasks**

The Novell eDirectory can be configured for use as a Tivoli Access Manager user registry in much the same way as any other LDAP directory. This section describes a few steps that are unique to this configuration.

Tasks to be performed include:

- Updating the eDirectory schema using the Novell eDirectory ConsoleOne directory management utility
- Novell eDirectory maintenance activities that can damage schema modifications applied by Tivoli Access Manager

**Updating the eDirectory schema**

If you are installing a new Tivoli Access Manager secure domain, the Tivoli Access Manager schema is installed on the Novell eDirectory Server (NSD) automatically when the Tivoli Access Manager policy server is configured. However, prior to configuring the policy server, there are several modifications to Novell eDirectory that must first be performed using Novell’s ConsoleOne directory management utility.

**Note:** The default Novell eDirectory schema assumes that the directory does not use the X.500 object classes of **inetOrgPerson** or **groupOfNames**. By default, these classes are mapped into the eDirectory classes of **User** and **Group**, respectively. Because Tivoli Access Manager uses the **inetOrgPerson** and **groupOfNames** object classes for creating its own users and groups, modifications to the default eDirectory schema are required.

Prior to configuring the Tivoli Access Manager policy server to automatically update the Novell eDirectory schema, do the following:
1. Start Novell’s ConsoleOne directory management utility.
2. Select the organization object within your Novell eDirectory tree. A list of objects appears on the right side of the ConsoleOne window.
3. Right click the LDAP group object (not LDAP server), and click Properties from the menu.
4. Click the Class Map tab and the table of LDAP class names. The Novell eDirectory class names are displayed.
5. Delete the entries with LDAP classes of inetOrgPerson and groupOfNames.
6. Click Apply, and then click Close.
7. Click the Attribute Map tab and the table of LDAP attribute names. The Novell eDirectory attribute names are displayed.
8. Scroll through the table and find the Novell eDirectory attribute member. Check the value of the corresponding LDAP attribute.
   - If the LDAP attribute is a member, no change is needed.
   - If the attribute is showing the default value of uniqueMember, you need to modify it.
9. Click Modify. The Attribute Mapping window is displayed.
10. Change the Primary LDAP Attribute field from uniqueMember to member.
11. Change the Secondary LDAP attribute field from member to uniqueMember.
12. Click OK. The Attribute Mapping window is cleared.
13. If you are using Solaris, proceed to step 18. If you are using Windows NT, you might have to add another mapping for the LDAP attribute ndsHomeDirectory.
14. To add another mapping for the LDAP attribute ndsHomeDirectory, click Add on the right hand side of the Attribute Mappings window. The Attribute Mapping window repaints and appears again.
15. From the Novell eDirectory NSD Attribute field menu, click Home Directory.
16. In the Primary LDAP Attribute field, click ndsHomeDirectory.
17. Click OK, and the Attribute Mapping window is cleared.
18. Click OK on the Properties dialog, and the Properties window is cleared.

**Novell eDirectory maintenance activities that can damage schema modifications applied by Tivoli Access Manager**

Novell eDirectory defines the object classes User and Group as part of its base schema. Instances of these object classes are created by an eDirectory administrator when defining a user or a group, respectively. Both of these object classes are defined by eDirectory as leaf nodes. eDirectory adds an attribute X-NDS_NOT_CONTAINER '1' to each of these object class definitions that specifies they are not container objects. Not being a container object means that the objects cannot be defined beneath instances of these object classes.

Tivoli Access Manager requires the ability to append its own objects beneath pre-existing eDirectory users and groups in order to import them and make them usable by Tivoli Access Manager. When Tivoli Access Manager adds its own object class definitions to the eDirectory schema, it also redefines the eDirectory User and Group object classes to allow instances of these classes to be container objects. Novell eDirectory allows this change to its schema definition.
The following Novell eDirectory administrator actions cause Tivoli Access Manager modification to the User object class to be undone. The Group object class is not affected.

- Running the eDirectory database repair tool ndsrepair using the rebuild schema option.
- Running Basic Repair from the iManager console and running local database repair with the rebuild operational schema option.
- Applying a patch update to Novell eDirectory.
- Upgrading Novell eDirectory to a more recent version.

Should it be necessary to perform any of these operations after Tivoli Access Manager has been configured into the eDirectory server, run the following Tivoli Access Manager command immediately to ensure that the definition of the User object class is restored.

```
ivrgy_tool(.exe) -h edir_server_name -p port -D edir_admin_dn -w edir_admin_password schema
```

The ivrgy_tool utility can be found in the following Tivoli Access Manager directory:

Windows:
c:\Program Files\Tivoli\Policy Director\sbin

UNIX:
/opt/PolicyDirector/sbin

Tivoli Access Manager does not add the sbin directory to the system PATH> You must run the ivrgy_tool utility from the sbin directory.
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
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, 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
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this 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.

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.
If you are viewing this information softcopy, the photographs and color illustrations may not appear.

**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
- OS/390
- SecureWay
- Tivoli
- Tivoli logo
- Universal Database
- WebSphere
- zSeries
- z/OS

Lotus and Domino are trademarks of International Business Machines Corporation and Lotus Development 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.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.
Glossary

A

access control. In computer security, the process of ensuring that the resources of a computer system can be accessed only by authorized users in authorized ways.

access control list (ACL). In computer security, a list that is associated with an object that identifies all the subjects that can access the object and their access rights. For example, an access control list is a list that is associated with a file that identifies the users who can access the file and identifies the users’ access rights to that file.

access permission. The access privilege that applies to the entire object.

action. An access control list (ACL) permission attribute. See also access control list.

ACL. See access control list.

administration service. An authorization API runtime plug-in that can be used to perform administration requests on a Tivoli Access Manager resource manager application. The administration service will respond to remote requests from the pdadmin command to perform tasks, such as listing the objects under a particular node in the protected object tree. Customers may develop these services using the authorization ADK.

attribute list. A linked list that contains extended information that is used to make authorization decisions. Attribute lists consist of a set of name = value pairs.

authentication. (1) In computer security, verification of the identity of a user or the user’s eligibility to access an object. (2) In computer security, verification that a message has not been altered or corrupted. (3) In computer security, a process that is used to verify the user of an information system or of protected resources. See also multi-factor authentication, network-based authentication, and step-up authentication.

authorization. (1) In computer security, the right granted to a user to communicate with or make use of a computer system. (2) The process of granting a user either complete or restricted access to an object, resource, or function.

authorization rule. See rule.

authorization service plug-in. A dynamically loadable library (DLL or shared library) that can be loaded by the Tivoli Access Manager authorization API runtime client at initialization time in order to perform operations that extend a service interface within the Authorization API. The service interfaces that are currently available include Administration, External Authorization, Credentials modification, Entitlements and PAC manipulation interfaces. Customers may develop these services using the authorization ADK.

B

BA. See basic authentication.

basic authentication. A method of authentication that requires the user to enter a valid user name and password before access to a secure online resource is granted.

bind. To relate an identifier to another object in a program; for example, to relate an identifier to a value, an address or another identifier, or to associate formal parameters and actual parameters.

blade. A component that provides application-specific services and components.

business entitlement. The supplemental attribute of a user credential that describes the fine-grained conditions that can be used in the authorization of requests for resources.

C

CA. See certificate authority.

CDAS. See Cross Domain Authentication Service.

CDMF. See Cross Domain Mapping Framework.

certificate. In computer security, a digital document that binds a public key to the identity of the certificate owner, thereby enabling the certificate owner to be authenticated. A certificate is issued by a certificate authority.

certificate authority (CA). An organization that issues certificates. The certificate authority authenticates the certificate owner’s identity and the services that the owner is authorized to use, issues new certificates, renews existing certificates, and revokes certificates belonging to users who are no longer authorized to use them.

CGI. See common gateway interface.
cipher. Encrypted data that is unreadable until it has been converted into plain data (decrypted) with a key.

cross modification. A multitude of attributes, such as authentication, organizations, container objects, and credentials, that are organized and interconnected. (2) The machines, devices, and programs that make up a system, subsystem, or network.

configuration. (1) The manner in which the hardware and software of an information processing system are established between functional units for conveying information. (2) In TCP/IP, the path between two protocol applications that provides reliable data stream delivery service. In the Internet, a connection extends from a TCP application on one system to a TCP application on another system. (3) In system communications, a line over which data can be passed between two systems or between a system and a device.

container object. A structural designation that organizes the object space into distinct functional regions.

cookie. Information that a server stores on a client machine and accesses during subsequent sessions. Cookies allow servers to remember specific information about clients.

credentials. Detailed information, acquired during authentication, that describes the user, any group associations, and other security-related identity attributes. Credentials can be used to perform a multitude of services, such as authorization, auditing, and delegation.

credentials modification service. An authorization API runtime plug-in which can be used to modify a Tivoli Access Manager credential. Credentials modification services developed externally by customers are limited to performing operation to add and remove from the credentials attribute list and only to those attributes that are considered modifiable.

cross domain authentication service (CDAS). A WebSEAL service that provides a shared library mechanism that allows you to substitute the default WebSEAL authentication mechanisms with a custom process that returns a Tivoli Access Manager identity to WebSEAL. See also WebSEAL.

cross domain mapping framework (CDMF). A programming interface that allows a developer to customize the mapping of user identities and the handling of user attributes when WebSEAL e-Community SSO function are used.

daemon. A program that runs unattended to perform continuous or periodic systemwide functions, such as network control. Some daemons are triggered automatically to perform their task; others operate periodically.

directory schema. The valid attribute types and object classes that can appear in a directory. The attribute types and object classes define the syntax of the attribute values, which attributes must be present, and which attributes may be present for the directory.

distinguished name (DN). The name that uniquely identifies an entry in a directory. A distinguished name is made up of attribute: value pairs, separated by commas.

digital signature. In e-commerce, data that is appended to, or is a cryptographic transformation of, a data unit and that enables the recipient of the data unit to verify the source and integrity of the unit and to recognize potential forgery.

dN. See distinguished name.

domain. (1) A logical grouping of users, systems, and resources that share common services and usually function with a common purpose. (2) That part of a computer network in which the data processing resources are under common control. See also domain name.

domain name. In the Internet suite of protocols, a name of a host system. A domain name consists of a sequence of subnames that are separated by a delimiter character. For example, if the fully qualified domain name (FQDN) of a host system is as400.rchland.vn.et.ribm.com, each of the following is a domain name: as400.rchland.vn.et.ribm.com, vn.et.ribm.com, ibm.com.

dE. See External Authorization Service.

encryption. In computer security, the process of transforming data into an unintelligible form in such a way that the original data either cannot be obtained or can be obtained only by using a decryption process.

entitlement. A data structure that contains externalized security policy information. Entitlements contain policy data or capabilities that are formatted in a way that is understandable to a specific application.

entitlement service. An authorization API runtime plug-in which can be used to return entitlements from an external source for a principal or set of conditions. Entitlements are normally application specific data that will be consumed by the resource manager application.
in some way or added to the principal’s credentials for use further on in the authorization process. Customers may develop these services using the authorization ADK.

**external authorization service.** An authorization API runtime plug-in that can be used to make application or environment specific authorization decisions as part of the Tivoli Access Manager authorization decision chain. Customers may develop these services using the authorization ADK.

**F**

**file transfer protocol (FTP).** In the Internet suite of protocols, an application layer protocol that uses Transmission Control Protocol (TCP) and Telnet services to transfer bulk-data files between machines or hosts.

**G**

**global signon (GSO).** A flexible single sign-on solution that enables the user to provide alternative user names and passwords to the back-end Web application server. Global signon grants users access to the computing resources they are authorized to use — through a single login. Designed for large enterprises consisting of multiple systems and applications within heterogeneous, distributed computing environments, GSO eliminates the need for users to manage multiple user names and passwords. See also **single signon.**

GSO. See global signon.

**H**

**host.** A computer that is connected to a network (such as the Internet or an SNA network) and provides an access point to that network. Also, depending on the environment, the host may provide centralized control of the network. The host can be a client, a server, or both a client and a server simultaneously.

**HTTP.** See **Hypertext Transfer Protocol.**

**hypertext transfer protocol (HTTP).** In the Internet suite of protocols, the protocol that is used to transfer and display hypertext documents.

**I**

**Internet protocol (IP).** In the Internet suite of protocols, a connectionless protocol that routes data through a network or interconnected networks and acts as an intermediary between the higher protocol layers and the physical network.

**Internet suite of protocols.** A set of protocols developed for use on the Internet and published as Requests for Comments (RFCs) through the Internet Engineering Task Force (IETF).

**interprocess communication (IPC).** (1) The process by which programs communicate data to each other and synchronize their activities. Semaphores, signals, and internal message queues are common methods of interprocess communication. (2) A mechanism of an operating system that allows processes to communicate with each other within the same computer or over a network.

**IP.** See **Internet Protocol.**

**IPC.** See **Interprocess Communication.**

**J**

**junction.** An HTTP or HTTPS connection between a front-end WebSEAL server and a back-end Web application server. WebSEAL uses a junction to provide protective services on behalf of the back-end server.

**K**

**key.** In computer security, a sequence of symbols that is used with a cryptographic algorithm for encrypting or decrypting data. See **private key** and **public key.**

**key database file.** See **key ring.**

**key file.** See **key ring.**

**key pair.** In computer security, a public key and a private key. When the key pair is used for encryption, the sender uses the public key to encrypt the message, and the recipient uses the private key to decrypt the message. When the key pair is used for signing, the signer uses the private key to encrypt a representation of the message, and the recipient uses the public key to decrypt the representation of the message for signature verification.

**key ring.** In computer security, a file that contains public keys, private keys, trusted roots, and certificates.

**L**

**LDAP.** See **Lightweight Directory Access Protocol.**

**lightweight directory access protocol (LDAP).** An open protocol that (a) uses TCP/IP to provide access to directories that support an X.500 model and (b) does not incur the resource requirements of the more complex X.500 Directory Access Protocol (DAP). Applications that use LDAP (known as directory-enabled applications) can use the directory as a common data store and for retrieving information about people or services, such as e-mail addresses, public keys, or service-specific configuration parameters. LDAP was originally specified in RFC
lightweight third party authentication (LTPA). An authentication framework that allows single sign-on across a set of Web servers that fall within an Internet domain.

LTPA. See lightweight third party authentication.

management domain. The default domain in which Tivoli Access Manager enforces security policies for authentication, authorization, and access control. This domain is created when the policy server is configured. See also domain.

management server. Obsolete. See policy server.

metadata. Data that describes the characteristics of stored data.

migration. The installation of a new version or release of a program to replace an earlier version or release.

multi-factor authentication. A protected object policy (POP) that forces a user to authenticate using two or more levels of authentication. For example, the access control on a protected resource can require that the users authenticate with both user name/password and user name/token passcode. See also protected object policy.

multiplexing proxy agent (MPA). A gateway that accommodates multiple client access. These gateways are sometimes known as Wireless Access Protocol (WAP) gateways when clients access a secure domain using a WAP. Gateways establish a single authenticated channel to the originating server and tunnel all client requests and responses through this channel.

network-based authentication. A protected object policy (POP) that controls access to objects based on the internet protocol (IP) address of the user. See also protected object policy.

P

PAC. See privilege attribute certificate.

permission. The ability to access a protected object, such as a file or directory. The number and meaning of permissions for an object are defined by the access control list (ACL). See also access control list.

policy. A set of rules that are applied to managed resources.

policy server. The Tivoli Access Manager server that maintains the location information about other servers in the secure domain.

polling. The process by which databases are interrogated at regular intervals to determine if data needs to be transmitted.

POP. See protected object policy.

portal. An integrated Web site that dynamically produces a customized list of Web resources, such as links, content, or services, available to a specific user, based on the access permissions for the particular user.

privilege attribute certificate. A digital document that contains a principal’s authentication and authorization attributes and a principal’s capabilities.

privilege attribute certificate service. An authorization API runtime client plug-in which translates a PAC of a predetermined format in to a Tivoli Access Manager credential, and vice-versa. These services could also be used to package or marshall a Tivoli Access Manager credential for transmission to other members of the secure domain. Customers may develop these services using the authorization ADK. See also privilege attribute certificate.

protected object. The logical representation of an actual system resource that is used for applying ACLs and POPs and for authorizing user access. See also protected object policy and protected object space.

protected object policy (POP). A type of security policy that imposes additional conditions on the operation permitted by the ACL policy to access a protected object. It is the responsibility of the resource manager to enforce the POP conditions. See also access control list, protected object, and protected object space.

protected object space. The virtual object representation of actual system resources that is used for applying ACLs and POPs and for authorizing user access. See also protected object and protected object policy.

private key. In computer security, a key that is known only to its owner. Contrast with public key.

public key. In computer security, a key that is made available to everyone. Contrast with private key.

Q

quality of protection. The level of data security, determined by a combination of authentication, integrity, and privacy conditions.
R

registry. The datastore that contains access and configuration information for users, systems, and software.

replica. A server that contains a copy of the directory or directories of another server. Replicas back up servers in order to enhance performance or response times and to ensure data integrity.

resource object. The representation of an actual network resource, such as a service, file, and program.

response file. A file that contains a set of predefined answers to questions asked by a program and that is used instead of entering those values one at a time.

role activation. The process of applying the access permissions to a role.

role assignment. The process of assigning a role to a user, such that the user has the appropriate access permissions for the object defined for that role.

routing file. An ASCII file that contains commands that control the configuration of messages.

RSA encryption. A system for public-key cryptography used for encryption and authentication. It was invented in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman. The system’s security depends on the difficulty of factoring the product of two large prime numbers.

rule. One or more logical statements that enable the event server to recognize relationships among events (event correlation) and to execute automated responses accordingly.

run time. The time period during which a computer program is executing. A runtime environment is an execution environment.

S

scalability. The ability of a network system to respond to increasing numbers of users who access resources.

schema. The set of statements, expressed in a data definition language, that completely describe the structure of a database. In a relational database, the schema defines the tables, the fields in each table, and the relationships between fields and tables.

secure sockets layer (SSL). A security protocol that provides communication privacy. SSL enables client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery. SSL was developed by Netscape Communications Corp. and RSA Data Security, Inc.

security management. The management discipline that addresses an organization’s ability to control access to applications and data that are critical to its success.

self-registration. The process by which a user can enter required data and become a registered Tivoli Access Manager user, without the involvement of an administrator.

service. Work performed by a server. A service can be a simple request for data to be sent or stored (as with file servers, HTTP servers, e-mail servers, and finger servers), or it can be more complex work such as that of print servers or process servers.

silent installation. An installation that does not send messages to the console but instead stores messages and errors in log files. Also, a silent installation can use response files for data input. See also response file.

single signon (SSO). The ability of a user to logon once and access multiple applications without having to logon to each application separately. See also global signon.

SSL. See Secure Sockets Layer.

SSO. See Single Signon.

step-up authentication. A protected object policy (POP) that relies on a preconfigured hierarchy of authentication levels and enforces a specific level of authentication according to the policy set on a resource. The step-up authentication POP does not force the user to authenticate using multiple levels of authentication to access any given resource but requires the user to authenticate at a level at least as high as that required by the policy protecting a resource.

suffix. A distinguished name that identifies the top entry in a locally held directory hierarchy. Because of the relative naming scheme used in Lightweight Directory Access Protocol (LDAP), this suffix applies to every other entry within that directory hierarchy. A directory server can have multiple suffixes, each identifying a locally held directory hierarchy.

token. (1) In a local area network, the symbol of authority passed successively from one data station to another to indicate the station temporarily in control of the transmission medium. Each data station has an opportunity to acquire and use the token to control the medium. A token is a particular message or bit pattern that signifies permission to transmit. (2) In local area networks (LANs), a sequence of bits passed from one device to another along the transmission medium. When the token has data appended to it, it becomes a frame.
trusted root. In the Secure Sockets Layer (SSL), the
public key and associated distinguished name of a
certificate authority (CA).

U

uniform resource identifier (URI). The character
string used to identify content on the Internet,
including the name of the resource (a directory and file
name), the location of the resource (the computer
where the directory and file name exist), and how the
resource can be accessed (the protocol, such as HTTP).
An example of a URI is a uniform resource locator, or
URL.

uniform resource locator (URL). A sequence of
characters that represent information resources on a
computer or in a network such as the Internet. This
sequence of characters includes (a) the abbreviated
name of the protocol used to access the information
resource and (b) the information used by the protocol
to locate the information resource. For example, in the
context of the Internet, these are abbreviated names of
some protocols used to access various information
resources: http, ftp, gopher, telnet, and news; and this
is the URL for the IBM home page:

URI. See uniform resource identifier.

URL. See uniform resource locator.

user. Any person, organization, process, device,
program, protocol, or system that uses a service
provided by others.

user registry. See registry.

V

virtual hosting. The capability of a Web server that
allows it to appear as more than one host to the
Internet.

W

Web Portal Manager (WPM). A Web-based graphical
application used to manage Tivoli Access Manager Base
and WebSEAL security policy in a secure domain. An
alternative to the pdadmin command line interface, this
GUI enables remote administrator access and enables
administrators to create delegated user domains and
assign delegate administrators to these domains.

WebSEAL. A Tivoli Access Manager blade. WebSEAL
is a high performance, multi-threaded Web server that
applies a security policy to a protected object space.
WebSEAL can provide single sign-on solutions and
incorporate back-end Web application server resources
into its security policy.
Index

A
access control list
See ACLs
access control list (ACL)
detaching from object 79
finding 79
access control lists
See ACLs
access decision information
See ADI
accessor output field 183
accountability 3
ACL commands
attach command 79
create command 77
delete command 77
detach command 80
find command 79
list command 77
modify command 78
show command 78
ACL policies 44
explicit 57
inheritance 57
ACL policies, defining 11
ACL policy administrator 43
ACLs 45
applying policies to different object types 57
applying to objects 78
attaching to object 78
control permission 55
creating 76
custom actions 84
custom actions example 85
default administration ACL 45
default config ACL 46
default domain ACL 46
default GSO ACL 46
default management ACL 46
default policy ACL 46
default proxy ACL 46
default replica ACL 46
default root ACL 45
definition of 12, 36
deleting 77
detaching from object 79
entering custom actions 85
entry syntax 73
entry type attribute 74
evaluating 38
explicit versus inherited 54
finding 79
group management policies 164
ID attribute 75
listing 77
management permissions 47
management tasks 76
managing ACL permissions 47
modifying 77
operations on an object 38
permissions 75
ACLs (continued)
permissions attribute 75
policies 38
policy entries 73
resolving request 56
showing 77
traverse permission 80
user management policies 165
versus authorization rules 40
action code events 182
action codes
authentication event 182
authorization event 182
change password event 182
management commands 182, 191
WebSEAL event 182
action commands
create command 84
action group
creating 82
creating new action in an action group 84
deleting 83
guidelines 82
listing 83
action group commands
create command 83
delete command 83
list command 83
action output field 182
actions 75
custom 84
custom, example 85
entering into ACL entries 85
managing Action permissions 48
activating
roles 158
add action 75
ADI
containers and container names 105
definition of 101
determining what is missing 118
dynamic retrieval entitlement services 118
name/value pair attributes 106
request from the resource manager 116, 117
retrieval entitlements service 103
source 103
XML document model 104
XML document model restriction 105
administrating
users and groups 125
administration
delegate role 157
enterprise domain 155
multiple domains 156
roles 157
superdomain 155
administration ACL (default) 45
administration API
Tivoli Access Manager component 4
administration groups 159

© Copyright IBM Corp. 1999, 2003
administration rights  
  gaining 44
administration users  
  creating 43
administration users and groups  
  defaults 43
administrator  
  multiple domains 156
  superdomain 155
  tasks 158
administrators  
  administrator 157
  domain 156
  enterprise domain 155
  Policy Director 156
  predefined 156
  sec_master 155
  senior 156
  support 157
  types 156
agents, log 197
algorithm, network-based authorization 96
allowed registry substring 153
any-authenticated type attribute  
  See any-other
any-other  
  authenticated requests 38
  type attribute 74
  unauthenticated requests 39
any-other user 38
app_context attribute list 102
application actions  
  execute (x) 76
  list directory (l) 75
  read (r) 75
application context information 102
applying  
  levels for step-up authentication 98
applying ACL policies 57
attach command 123
attaching  
  ACLs to objects 78
attaching authorization rule 122
attribute lists  
  app_context 102
  permission_info_returned 117
attribute name, XML 106
attribute output field 187
attribute pairs (name/value) 101
attributes  
  ACL entry types 74
  azn_cred_groups 114
  azn_cred_principal_name 114
  azn_cred_registry_id 114
  azn_engine_requested_actions_attribute 103
  azn_engine_target_resource 103
  azn_init_set_perminfo_attrs 116
  azn_perminfo_reason_rule_failed 117
  azn_perminfo_rules_adi_request 116, 118
POPs 87
audit event, definition 225
audit events  
  capturing 197
  configuring console log agent 202
  configuring EventPool 198
  configuring file log agent 203
  configuring pipe log agent 207
audit events (continued)  
  configuring remote log agent 208
  event categories 197
  log agents 197
audit file buffers 227
audit output field 190
audit records  
  authentication 221
  authorization 220
  management 223
audit tag 225
audit trail file 225
audit trail files 227
  contents 220
  flushing buffers 227
  format 179
audit trail, definition 225
audit.log file 226
auditcfg parameter 225
auditing  
  enabling and disabling 225
  introduction 219
  overview 169
auditlog logging parameter 226
authenticated  
  authenticated requests 38
  unauthenticated requests 39
authenticated requests 38
authentication  
  applying step-up policy 98
  audit records 221
  configuring levels of step-up 97
  multi-factor authentication 98
Tivoli Access Manager component 2
authentication-mechanisms stanza 242
authorization  
  audit records 220
  evaluation process 18
  step-by-step process 12
Tivoli Access Manager component 2
authorization API  
  introducing 13
  local cache mode 16
  remote cache mode 15
  standard 5
authorization API attributes 101
authorization database  
  replicating 145
authorization evaluator  
  overview 9
authorization model  
  conceptual model 5
authorization policy database 8
Authorization rule language  
  languages 104
authorization rules 11, 12
  attaching to an object 122
  changing 122
  creating 120
  definition of 37
  deleting 121
  delimiters 110
  detaching 124
  differing from ACLs and POPs 40
  evaluator 110
  examples 113
  finding 123
authorization rules (continued)  
  Format and constraints 111  
  listing 121  
  managing 101, 120  
  managing rule permissions 53  
  overview 101  
  policies 40  
  policies for 59  
  when to use 40  
authorization server  
  pdacd daemon 139  
  preventing automating startup 145  
authorization service 6  
  authorization API 9  
  benefits 7  
  extending 17  
  interfaces 9  
  overview 8  
Tivoli Access Manager 7  
authzrule attach command 123  
authzrule commands 120  
  attach command 123  
  create command 121  
  delete command 122  
  detach command 124  
  find command 123  
  list command 121  
  modify command 122  
authzrule create command 121  
authzrule delete command 122  
authzrule detach command 124  
authzrule find command 123  
authzrule list command 121  
authzrule modify command 122  
auto-database-update-notify 146  
auto-database-update-notify command 145  
automatic certificate and password refresh 135  
anz_output_field 185  
anz_cred_groups attribute 114  
anz_cred_principal_name attribute 114  
anz_cred_registry_id attribute 114  
anz_decision_access_allowed_ext () call 102  
anz_decision_access_allowed_ext() call 116, 117  
anz_decision_access_allowed() call 116  
anz_engine_requested_actions attribute 103  
anz_engine_target_resource attribute 103  
anz_entitlement_get_entitlements() call 111, 118  
anz_init_set_perminfo_attrs attribute 116  
anz_init_set_perminfo_attrs initialization parameter 117  
anz_perminfo_reason_rule_failed attribute 117  
anz_perminfo_rules_adl_request attribute 116, 118  
anznapi-configuration stanza 247  
anznapi-cred-modification-services stanza 258  

B  

Base actions  
  add (A) 75  
  browse (b) 75  
  bypass POP (B) 75  
  control (c) 75  
  create (N) 75  
  delegate (g) 75  
  password (W) 76  
  trace (t) 76  
  traverse (T) 76  
basslcfg –config command 137  

benefits of authorization service 7  
blades 35  
boot-start-ivacld command 145  
boot-start-ivmgrd command 145  
boot-start-pdmgproxyd command 145  
browse action 75  
buffer size of log files 204, 209  
buffer_size 209  
buffer_size logging parameter 204, 209  
buffers  
  compressing for messages 209  
  flushing audit file buffers 227  
  flushing consolidation buffer 209  
  flushing log file buffers 202, 206, 208  
bypass POP action 75  

C  

CA  
  See certificate authority  
  cache file location 210  
  capturing audit events 225  
  categories of logging events 197  
  centralized management 3  
  certificate authority  
  See also CA  
  definition 133  
  determining trust 136  
  certificates  
    automatic refresh 135  
    initial configuration 134  
    management tasks 133  
    revocation 136  
    tools for updating 135  
  changing  
    ACLs 77  
    certificates 136  
    password 136  
  changing authorization rule 122  
  chgcert option of svrsslcfg command 136  
  chgpwd option of svrsslcfg command 136  
commands  
  authzrule attach 123  
  authzrule create 121  
  authzrule delete 122  
  authzrule detach 124  
  authzrule find 123  
  authzrule list 121  
  authzrule modify 122  
  basslcfg –config 137  
  for POPs 87  
  group list 166  
  group list-dn 166  
  pdadmin acl attach 79  
  pdadmin acl create 77  
  pdadmin acl delete 77  
  pdadmin acl detach 80  
  pdadmin acl find 79  
  pdadmin acl modify 78  
  pdadmin acl show 78  
  pdadmin action create 84  
  pdadmin action group create 83  
  pdadmin action group delete 83  
  pdadmin action group list 83  
  pdadmin domain create 62  
  pdadmin domain delete 63
commands (continued)
pdadmin domain list 62
pdadmin domain modify 64
pdadmin errtext 181
pdadmin group create 131, 132
pdadmin group create command 163
pdadmin group import 132
pdadmin object create 67, 69
pdadmin object create command 161
pdadmin object delete 71
pdadmin object list 70
pdadmin objectspace create 66
pdadmin objectspace delete 68
pdadmin objectspace list 68
pdadmin pop attach 92
pdadmin pop create 90
pdadmin pop delete 90
pdadmin pop detach 93
pdadmin pop find 92
pdadmin pop list 90
pdadmin pop modify 91
pdadmin pop show 91
pdadmin server replicate 146
pdadmin server task 217
pdadmin user create 126, 127, 128, 129
pdadmin user list 131
pdadmin user modify password 130
stats get 217
stats list 217
svrsslcfg -chgcert 136
svrsslcfg -chgpwd 136
svrsslcfg -unconfig 136
user list 166
user list-dn 166
user show groups 166
component output field 182
compress logging parameter 209
config ACL (default) 46
config option of basslcfg command 137
configuration
server files 144
configuration entries
dynamic-adi-entitlement-services 118
input-adi-xml-prolog 119
resource-manager-provided-adi 118
xsl-stylesheet-prolog 119
configuration files
default location 231
ldap.conf 287
configuration-database stanza 266
configurations
managing Config permissions 50
configuring
audit tag events 225
basslcfg -config command 137
central event propagation queue 201
certificates 134
console log agent 202, 203
event queue high water mark 201
EventPool category 198
file mode 207
frequency for flushing audit file buffers 227
frequency for flushing consolidation buffer 209
frequency for flushing log file buffers 202, 206, 208
high water mark 208
initialization attributes 118
levels for step-up authentication 97
configuring (continued)
location of program receiving log output 207
maximum number of events to queue 201
pipe log agent 207
remote log agent 208
console log agent 202
console logging 202
consolidation buffer 209
constraints, authorization rules 111
container names
  for ADI containers 105
  limitations of 105
container objects 34, 161
containers for ADI 105
contents, audit trail file 220
context information
  application 102
  authorization engine 103
control action 75
control permission 55
core technologies 2
count() function 115
create action 75
create command 121
creating
ACLs 76
action group 82
authorization rule 120
group container objects 161
new action in an action group 84
object 66
object space 65
objects 69
POP 88
roles 158
credential entitlements 102
custom actions 85

data integrity or redundancy 149
data output field 190
database
  path for user registry 153
date output field 180
default
  administration ACL 45
  administration users and groups 159
  config ACL 46
domain ACL 46
GSO ACL 46
management ACL 46
policy ACL 46
proxy ACL 46
replica ACL 46
root ACL 45, 55
security policy 43
default proxy ACL 46
defining
  security policy 44
delegate action 75
delegate administrator, illustrated 156
delegate role
  administration 157
delegated administration
  administration users and groups 43
group ACL permissions 164
files
audit.log 226
find command 123
finding authorization rules 123
flushing
audit file buffers 227
consolidation buffer 209
log file buffers 202, 206, 208
forbidden access to identifies 96
formats, authorization rules 111
formats
audit events 179
frequency for flushing
audit file buffers 227
consolidation buffer 209
log file buffers 202, 206, 208
G
generic actions
delete (d) 75
modify (m) 75
server administration (s) 76
view (v) 76
gid output field 186
Global Sign-On
See GSO
grant access to identifies 96
group
creating 131
importing 132
iv-admin 36
listing 130
searching 130
group commands
create 163
create command 131, 132
import command 132
group container objects
creating 161
group list command 166
group list-dn command 166
group management 164
groups
creating 163
definition of 125
debating management 160
iv-admin 36
iv-admin group 43
ivmgrd-servers 43
managing groups permissions 52
groups
type attribute 74
GSO
managing GSO permissions 53
GSO ACL (default) 46
guidelines
for secure object space 41
key and stash file renewal 137
H
HACMP 149
guidelines for cluster 149
hi_water logging parameter 206, 208, 209
hi_water parameter 201
High Availability Cluster Multiprocessing
See HACMP
high water mark 209
high water mark event queue 201, 206, 208
I
IBM Directory 339
IBM Directory user registry
See LDAP server
IBM products
IBM Tivoli Access Manager 1
IBM Tivoli Access Manager for Business Integration 1
IBM Tivoli Access Manager for Operating Systems 1
IBM Tivoli Access Manager WebSEAL 1
IBM Tivoli Access Manager
WebSEAL component 1
IBM Tivoli Access Manager for Business Integration 1
IBM Tivoli Access Manager for Operating Systems 1
ID of log files 203
inherited ACL 54, 57
inherited policy 11
initialization attributes 118
dynamic-adi-entitlement-services 118
input-adi-xml-prolog 119
resource-manager-provided-adi 118
xsl-stylesheet-prolog 119
input-adi-xml-prolog configuration entry 119
integrity of data 149
IP addresses
disabling network-based authorization for POPs 96
setting for POPs 95
iPlanet
See Sun ONE
iv-admin group 36, 43
ivacld stanza 270
ivacld.kdb key file 135
ivacld.sth stash file 135
ivmgrd stanza 273
ivmgrd-servers group 43
ivmgrd.kdb key file 135
ivmgrd.sth stash file 135
K
key files
definition 133
ivacld.kdb 135
ivmgrd.kdb 135
pd.kdb 135
pdmgrproxyd.kdb 135
L
LDAP
configuration 287
configuration file 278
failover configuration 339
LDAP failover
preference values 341
management
  audit records 223
management ACL (default) 46
management commands 191
management delegation 158, 160
management domain 33
management interface 9
management region
  permissions 47
management tasks
  authorization rules 120
Management/ACL permissions 47
management/Action permissions 48
Management/Config permissions 50
Management/Domain permissions 54
Management/Groups permissions 52
Management/GSO permissions 53
Management/Policy permissions 50
Management/POP permissions 49
Management/Proxy permissions 54
Management/Replica permissions 50
Management/Rule permissions 53
management/Server permissions 49
Management/Users permissions 51
managing
  ACL permissions 47
  Action permissions 48
  authorization rules 101
certificates 133
  Config permissions 50
delegated administration policy 166
domain permissions 54
groups permissions 52
GSO permissions 53
  object space 65
  objects 69
  passwords 133
  policy permissions 50
POP permissions 49
proxy permissions 54
replica permissions 50
rule permissions 53
  server permissions 49
  servers 139
  users permissions 51
mapping, pdadmin CLI to WPM 329
master authorization policy database 8
max-notifier-threads 146
max-notifier-threads stanza entry 147
maximum number of events to queue 205, 207
messages
  compressing buffers 209
  messages, serviceability 169
meta-info stanza 294
mode logging parameter 207
model of authorization 5
model, document 104
modes
  authorization API local cache 16
  authorization API remote cache 15
text or binary files 207
modify action 75

Index 373
modify command 122
MQSeries 1
msg__error.log 170
msg__fatal.log 170
msg__notice.log 170
msg__verbose 170
msg__warning 170
multi-factor authentication 98
multiple domain 156
example 156
illustrated 156
multiple-tenancy policy server 153

N
name output field 186, 188
network-based authorization algorithm 96
disabling for POPs 96
limitations 96
network-based authorization by IP address 96
notice log file messages 170
tnotices 357
notification delay time 147
notifier-wait-time 146
notifier-wait-time command 147

O
object
creating 66
object commands
create 67, 69, 161
delete 71
list 70
specifying types 70
object create 67
object create command 69
object delete command 71
object list command 70
object output field 184
object permissions 81
object space
creating 65
definition of 34
delegating management 158
deleting 68
listing 67
management tasks 65
types 66
object space permissions 81
object types 67, 162
objects
creating 69
deleting 70
listing 70
objectspace commands
create 66
delete 68
list 68
objectspace create 66
objectspace delete 68
objectspace list 68
originator output field 181
outcome output field 181
output fields 179
parameters
auditcfg 225
hi_water 201
logcfg 201
logflush 227
logsize 227
queue_size 201
password changing 130
troubleshooting 157
password action 76
passwords
automatic refresh 135
management tasks 133
path logging parameter 203, 207, 210
path option for logging 203
pd.kdb key file 135
pd.sth stash file 135
pdacl 139
key and stash files 135
padmin action create command 84
padmin action group create command 83
padmin action group delete command 83
padmin action group list command 83
padmin errtext command 181
padmin group create command 163
padmin object create 67
padmin object create command 69, 161
padmin object delete 71
padmin object list 70
padmin objectspace create 66
padmin objectspace delete 68
padmin objectspace list 68
padmin pop attach command 92
padmin pop create command 90
padmin pop delete command 90
padmin pop detach command 93
padmin pop find command 92
padmin pop list command 90
padmin pop modify command 91
padmin pop show command 91
padmin server
replicating 146
padmin server replicate 146
padmin server replicate command 146
padmin server task command 217
padmin stats get command 217
padmin stats list command 217
padmin utility
Tivoli Access Manager component 4
PDCA certificate
See certificates
pdmgrd 139
key and stash files 135
pdmgrproxyd 139
key and stash files 135
pdmgrproxyd stanza 295
pdmgrproxyd.kdb key file 135
pdmgrproxyd.sth stash file 135
PDRT 135
pdrte stanza 298
pdwpm stanza 301
performance 217
perm output field 185
permission_info attribute list 117
permissions 75
permissions (continued)
/Management/ACL permissions 47
/management/Action permissions 48
/Management/Config permissions 50
/Management/Domain permissions 54
/Management/Groups permissions 52
/Management/GSO permissions 53
/Management/Policy permissions 50
/Management/POP permissions 49
/Management/Proxy permissions 54
/Management/Replica permissions 50
/Management/Rule permissions 53
/Management/Server permissions 49
/Management/Users permissions 51
administration users 43
control (c) 55
custom 84
iv-admin group 43
object 81
object space 81
roles 157
sec_master 43
traverse (T) 55, 80
pid output field 186
pipe log agent 207
pipe logging parameters 207
policies
ACL 44
authorization rules 40, 59
managing policy permissions 50
POPs 39
policy ACL (default) 46
policy database 8
Policy Director
administrators 156
policy enforcer 6
policy output field 186
policy server
See also pdmgrd
pdmgrd daemon 139
preventing automating startup 145
POP
applying to objects 91
attaching to object 92
configuring attributes 93
creating 88
deleting 90
detaching from object 93
disabling network-based authorization by IP address 96
finding where attached 92
introducing 87
listing 90
modifying 91
showing 91
specifying IP addresses and ranges 95
pop attach command 92
POP attributes
audit level 93
time of day 94
warning mode 93
POP attributes enforcement
by resource manager 59
by Tivoli Access Manager Base 58
POP commands
attach 92
create 90
delete 90
POP commands (continued)
detach 93
find 92
list 90
modify 91
show 91
pop create command 90
pop delete command 90
pop detach command 93
pop find command 92
pop list command 90
pop modify command 91
POP policies, defining 11
POP policy
network-based authorization 95
quality of protection 99
pop show command 91
POPs
attributes of 58
definition of 12, 36
managing POP permissions 49
policies 39
versus authorization rules 40
port logging parameter 210
port, remove server 210
preference values (LDAP failover) 341
preventing automating startup
authorization server 145
policy server 145
proxy server 145
principal output field 183
process output field 186
prolog statements 119
propagation queue 217
protected object policies
See POPs
protected object policy
See POPs
protected object space 34
guidelines 41
protected objects 78
management tasks 69
Protected Resource Administrator 44
proxies
managing proxy permissions 54
proxy ACL (default) 46
proxy server
introducing 140
pdmgrproxyd daemon 139
preventing automating startup 145
Q
qualifier output field 185
quality of protection 2
queue high water mark 201, 206, 208
queue in memory 201
queue_size logging parameter 205, 207, 209
queue_size parameter 201
R
read action 75
reason codes
rule failure 117
rebind logging parameter 210
rebind retry timeout 210
redundancy of data 149
refresh, certificates and passwords 135
related publications xv
remote cache mode 13, 15
remote log agent 208
remote logging parameter 209
remote logging parameters 208
remote server
  buffer size before relaying 209
  cache file location 210
  consolidating into large buffers 209
  distinguished name 210
  host name 210
  nominating the hosts 210
  port 210
  rebinding retry timeout 210
  sending events to 208
remove access to identifies 96
renamelogs, key and stash files 135
replica 341
replica ACL (default) 46
replicas
  managing replica permissions 50
replicate authorization database 145
replicate pdadmin server 146
replication 9
resolving ACL request 56
resource manager 6
  enforcement of POP attributes 59
  requesting ADI 116
resource manager servers 134
resource objects 34
resource requests 18
resource-manager-provided-adi configuration entry 118
responsibilities
  Deployment Administrator 44
Protected Resource Administrator 44
Security Policy Administrator 44
restarting servers 143
restriction, ADI XML document model 105
result output field 185
retrieval entitlement services 118
retrieval entitlements service 103
retry timeout 210
retry timeout value 210
roles
  assigning role assignment 158
  defined 157
  delegate 157
  permissions 157
  role activation 158
  role assignment 158
  role creation 158
rollover thresholds 227
rollover size logging parameter 204
root ACL (default) 45, 55
rule failure reason code 117
security policy 36
  default 43
  defining 44
  implementing 11
overview 4
Security Policy Administrator 44
server administration action 76
server replicate 146
server status using the pd_start utility 143
server task command 217
servers
  authorization server 143, 149
  automating startup 145
  capturing audit events 225
  configuration files 144
  dependencies for 141
  distinguished name for remote 210
  HACMP cluster 149
  host name for remote 210
  management of 139
  managing server permissions 49
  policy proxy server 143
  policy server 142
  port for remote 210
  primary and standby 149
  resource manager 134
  restarting using the pd_start utility 143
  starting and stopping for UNIX 142
  starting and stopping for Windows 143
  starting in order manually 142
  starting using the pd_start utility 142
  starting using the Windows Services Control Panel 143
  stopping using the pd_start utility 143
serviceability messages 169
set ipauth add option 95, 96
set ipauth forbidden option 96
set ipauth remove option 96
shared disk arrays 149
size of log files 204
source output field 188
sources for ADI 103
sparse ACL model 54
ssl stanza 303, 309
standard error (stderr) for logging 203
standard out (stdout) for logging 202
stanzas
  authentication-mechanisms 242
  aznapic-configuration 247
  aznapic-cred-modification-services 258
  configuration-database 266
  delegated-admin 267
  domains 268
  icmdld 270
  ivmgrd 273
  ldap 278, 287
  manager 292
  meta-info 294
  pdmgrproxyd 295
  pdtde 298
  pdwpm 301
  ssl 303, 309
  uraf-registry 310
  uraf-registry in activedir_ldap.conf 318
  uraf-registry in activedir.conf 316
  xmladi-attribute-definition 323
starting servers
  manually in order 142
Index

starting servers (continued)
UNIX 142
using the pd_start utility 142
using the Windows Services Control Panel 143
Windows 143

stash files
definition 134
ivacld.sth 135
ivmgrpdx.sth 135
pd.sth 135
pdmgrproxydx.sth 135
stats get command 217
stats list command 217
status, server 143
step-up authentication
applying 98
configuring 97
versus multi-factor authentication 98

stopping servers
UNIX 142
using the pd_start utility 143
using the Windows Services Control Panel 143
Windows 143
string identifiers 111
subdomain 155
sum() function 115
superdomain 155
svrsslcf -chgpwd command 136
svrsslcf -unconfig command 136

tag-value support 102
target output field 184
tasks
role activation 158
role administration 158
role assignment 158
role creation 158
roles 157
types 158
timeout value, error retry 210
timeout, rebind retry 210
Tivoli Access Manager
accountability 3
administration API 4
audit trail files 225
authentication 2
authorization 2
authorization API 13
authorization service 7, 8
centralized management 3
core technologies 2
introduction to 1
pdadmin command line utility 4
quality of protection 2
restarting servers using the pd_start utility 143
scalability 3
security policy 157
starting server in order manually 142
starting servers using the pd_start utility 142
starting servers using the Windows Services Control Panel 143
stopping servers using the pd_start utility 143
stopping servers using the Windows Services Control Panel 143
utilities 142

Tivoli Access Manager (continued)
Web Portal Manager 4
Tivoli Access Manager Base
enforcement of POP attributes 58
tools for manual certificate update 135
trace action 76
trace events 178
trademarks 359
traverse 80
traverse action 76
traverse permission 55, 80
type attributes 74
type output field 187, 189
types of
object space 66
objects 67, 70
Tivoli Access Manager servers 139
Tivoli Access Manager utilities 142

U
uid output field 186
unauthenticated
type attribute 74
unauthenticated requests 39
unauthenticated user 38
UNIX servers
starting and stopping 142
update-notifier threads 147
uraf-registry stanza 310
uraf-registry stanza in activedir_ldap.conf 318
uraf-registry stanza in activedir.conf 316
user
creating 126
credential entitlements 102
importing 127
searching 125
sec_master 36
user commands
create command 126, 127, 128, 129
list command 131
modify password command 130
user list command 166
user list-dn command 166
user management 165
user registry
differences 325
LDAP configuration file 278, 287
maximum values 326, 327
user show groups command 166
users
administrator 36
administrator, administrator 157
administrator, domain 156
administrator, Policy Director 156
administrator, sec_master 155
administrator, senior 156
administrator, support 157
definition of 125
delegate 157
managing users permissions 51
sec_master 43
type attribute 74

Index 377
V

value output field 189
verbose log file messages 170
view action 76

W

warning log file messages 170
water mark, event queue 201, 206, 208
Web Portal Manager
  Tivoli Access Manager component 4
WebSEAL 1
Windows
  starting and stopping servers 143

X

XML
  ADI containers and container names 105
  ADI document model 104
  ADI document model restriction 105
  ADI name/value pair attributes 106
  audit trail output fields 179
  authorization rules 104
  entitlement document example 107
  prolog statements 119
xmladi attribute-definition stanza 323
XSL
  authorization rules 104
  prolog statements 119
  xsl:template statement 113
  xsl:when statement 114
xsl-stylesheet-prolog configuration entry 119
xsl:template statement 113
xsl:when statement 114