

# **Administrator's Guide**

Netscape Certificate Management System

Version 4.1

Netscape Communications Corporation ("Netscape") and its licensors retain all ownership rights to the software programs offered by Netscape (referred to herein as "Software") and related documentation. Use of the Software and related documentation is governed by the license agreement for the Software and applicable copyright law.

Your right to copy this documentation is limited by copyright law. Making unauthorized copies, adaptations, or compilation works is prohibited and constitutes a punishable violation of the law.

Netscape may revise this documentation from time to time without notice.

THIS DOCUMENTATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IN NO EVENT SHALL NETSCAPE BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY KIND ARISING FROM ANY ERROR IN THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION ANY LOSS OR INTERRUPTION OF BUSINESS, PROFITS, USE, OR DATA.

The Software and documentation are copyright ©1999 Netscape Communications Corporation, a subsidiary of AmericaOnline, Inc. All rights reserved.

Portions of the Software copyright © 1994-1995 Sun Microsystems, Inc. All rights reserved.

Netscape, Netscape Navigator, Netscape Certificate Server, Netscape DevEdge, Netscape FastTrack Server, Netscape ONE, SuiteSpot, and the Netscape N and Ship's Wheel logos are registered trademarks of Netscape Communications Corporation in the United States and other countries. Other Netscape logos, product names, and service names are also trademarks of Netscape Communications Corporation, which may be registered in other countries. JavaScript is a trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape. Other product and brand names are trademarks of their respective owners.

The downloading, exporting, or reexporting of Netscape software or any underlying information or technology must be in full compliance with all United States and other applicable laws and regulations. Any provision of Netscape software or documentation to the U.S. Government is with restricted rights as described in the license agreement accompanying Netscape software.



Recycled and Recyclable Paper

#### The Team:

Engineering: Andrew Wnuk, Christina Fu, Christine Ho, James Nicolson, John Hines, Lily Hsiao, Matthew Harmsen, Michelle Zhao, Ross Fubini, Steve Parkinson, Terry Hayes, Thomas Kwan

Marketing: Shirly March

Publications: Carol Henderson, Supriya Shetty

Quality Assurance: Ben Scharp, Beomsuk Kim

Release Engineering: Walt Miller

Version 4.0

Part Number: 151-09679-00

© Copyright 1999 Netscape Communications Corp., a subsidiary of AmericaOnline, Inc. All rights reserved  
All Rights Reserved

Printed in USA

99 98 97 10 9 8 7 6 5 4 3 2 1

# Contents

<b>About This Guide</b> .....	23
What's in This Guide .....	23
Who Should Read This Guide .....	23
What You Should Already Know .....	24
Conventions Used in This Guide .....	25
Where to Go for Related Information .....	27

## **Part I Netscape Certificate Management System**

<b>Chapter 1 Introduction to Certificate Management System</b> .....	31
Overview .....	32
Key Features .....	33
System Architecture .....	39
LDAP Directory Integration .....	42
How the Main Subsystems Function .....	43
Entry Points for Various Types of Users .....	45
<b>Chapter 2 Administration Tasks and Tool</b> .....	47
Netscape Console .....	48
Console Tab .....	49
Users and Groups Tab .....	50
Netscape Administration Server .....	51
Starting Administration Server .....	52
Shutting Down Administration Server .....	53
Accessing Netscape Console .....	53
The CMS Window .....	55
Tasks Tab .....	57
Configuration Tab .....	58
Status Tab .....	63

Accessing the CMS Window .....	63
<b>Chapter 3 Configuration</b> .....	67
Effects of Installation Type on Configuration .....	68
Duplicating a Configuration from One Instance to Another .....	70
Locating the Configuration File .....	71
Modifying the Configuration .....	71
Changing the Configuration from the CMS Window .....	71
Changing the Configuration by Editing the Configuration File .....	72
Guidelines for Editing the Configuration File .....	73
Sample Configuration File .....	74
Road Map to Configuring Subsystems .....	83

## **Part 2 Managing Certificate Management System**

<b>Chapter 4 Installing and Uninstalling Instances</b> .....	93
Installing Multiple Instances .....	94
Viewing Instance Information .....	96
Changing the Name of an Instance .....	98
Removing an Instance from a System .....	99
Uninstalling Certificate Management System .....	101
Uninstalling from the Command Line .....	101
Uninstalling by Using the Windows NT Add/Remove Programs Utility ...	102
<b>Chapter 5 Starting and Stopping Instances</b> .....	105
Starting Certificate Management System .....	106
Required Start-up Information .....	106
Starting from Netscape Console .....	107
Starting from the Command Line .....	109
Starting from the Windows NT Services Panel .....	109
Stopping Certificate Management System .....	110
Stopping from Netscape Console .....	111
Stopping from the Command Line .....	112
Stopping from the Windows NT Services Panel .....	112
Restarting Certificate Management System .....	113
Restarting from the CMS Window .....	113

Restarting from the Command Line .....	115
Checking System Status .....	115
Attending to an Unresponsive Server .....	116
CMS Watchdog Process .....	117

## Part 3 System-Level Configuration

<b>Chapter 6 Configuring Ports, Database, and SMTP Settings .....</b>	<b>121</b>
CMS Ports .....	121
Remote Administration Port .....	122
Agent Port .....	123
End-Entity Ports .....	124
Configuring Port Numbers .....	125
Specifying IP Addresses for CMS Instances .....	127
Internal Database .....	128
Configuring the Internal Database .....	129
SMTP Settings .....	131
<b>Chapter 7 Managing Privileged Users and Groups .....</b>	<b>133</b>
Privileged-User Types and Responsibilities .....	134
Administrators .....	134
Agents .....	135
Agent's Certificate for SSL Client Authentication .....	137
Trusted Managers .....	141
Subsystems That Can Function as Trusted Managers .....	141
Connectors for Linking Trusted Managers .....	143
Trusted Manager's Certificate for SSL Client Authentication .....	144
Groups and Their Privileges .....	146
Group for Administrators .....	146
Groups for Agents .....	147
Group for Certificate Manager Agents .....	147
Group for Registration Manager Agents .....	148
Group for Data Recovery Manager Agents .....	148
Group for Trusted Managers .....	149

Setting Up Privileged Users .....	149
Setting Up Administrators .....	150
Step 1. Find the Required Information .....	150
Step 2. Add the Information to the Internal Database .....	151
Setting Up Agents .....	152
Step 1. Find the Required Information .....	153
Step 2. Add the Information to the Internal Database .....	154
Step 3. Store the Agent's SSL Client Certificate in the Internal Database .....	156
Step 4. Check the Certificate Database for the CA Certificate .....	158
Setting Up Trusted Managers .....	158
Setting Up a Registration Manager as a Trusted Manager .....	158
Setting Up a Certificate Manager as a Trusted Manager .....	167
Changing Privileged-User Information .....	175
Changing a Privileged User's Login Information .....	176
Changing a Privileged User's Certificate .....	178
Changing Members in a Group .....	179
Deleting a Privileged User .....	181
<b>Chapter 8 Keys and Certificates .....</b>	<b>183</b>
Keys and Certificates for the Main Subsystems .....	184
Certificate Manager's Key Pairs and Certificates .....	184
CA Signing Key Pair and Certificate .....	184
SSL Server Key Pair and Certificate .....	186
Registration Manager's Key Pairs and Certificates .....	187
Signing Key Pair and Certificate .....	188
SSL Server Key Pair and Certificate .....	188
Data Recovery Manager's Key Pairs and Certificates .....	190
Transport Key Pair and Certificate .....	190
Storage Key Pair .....	191
SSL Server Key Pair and Certificate .....	191
Tokens for Storing Keys and Certificates .....	192
Internal Token .....	193
External Token .....	193
Installing External Tokens .....	194

Managing Tokens Used by the Subsystems .....	197
Viewing Tokens .....	197
Changing a Token's Password .....	198
Hardware Cryptographic Accelerators .....	198
Certificate Setup Wizard .....	199
Using the Wizard to Request a Certificate .....	200
Step 1. Select the Operation .....	201
Step 2. Choose the Certificate .....	202
Step 3. Specify the Key-Pair Information .....	204
Step 4. Specify the Subject Name for the Certificate .....	206
Step 5. Specify the Validity Period .....	207
Step 6. Specify Extensions .....	208
Step 7. Copy the Certificate Signing Request .....	210
Step 8. Check the Certificate Request Status .....	211
Step 9. Send the Certificate Signing Request to a CA .....	212
Using the Wizard to Install a Certificate or Certificate Chain .....	215
Data Formats for Installing Certificates and Certificate Chains .....	216
Step 1. Select the Operation .....	218
Step 2. Select the Certificate or Certificate Chain .....	219
Step 3. Specify the Location of the Certificate .....	220
Step 4. View the Certificate or Certificate Chain .....	222
Step 5. Install the Certificate or Certificate Chain .....	222
Step 6. Verify the Certificate Status .....	223
Configuring the Server's Security Preferences .....	223
Configuring the Server to Use Separate SSL Server Certificates .....	224
Step 1. Get the Required SSL Server Certificates .....	224
Step 2: Update the Configuration .....	225
Getting an SSL Client Certificate for a Subsystem .....	226
Step 1. Generate a Key Pair for the Subsystem .....	226
Step 2. Generate a Certificate Signing Request for the Key Pair .....	226
Step 3. Submit the CSR to the CA .....	227
Step 4. Ask an Agent to Approve the Request .....	227
Step 5. Install the Certificate in the Internal Database .....	227

Step 6. Configure the Subsystem to Use This Certificate .....	227
Setting Up Cipher Preferences for SSL Communications .....	228
SSL Ciphers Supported in Certificate Management System .....	228
Configuring the Server to Use Specific Ciphers .....	230
Getting New Certificates for the Subsystems .....	232
Step 1. Plan for the New Certificate .....	232
Step 2. Request the New Certificate .....	235
Step 3. Install the New Certificate .....	235
Step 4. Deploy the New Certificate .....	235
Deploying Certificate Manager's CA Signing Certificate .....	236
Deploying Registration Manager's Signing Certificate .....	236
Deploying Data Recovery Manager's Transport Certificate .....	237
Deploying a Subsystem's SSL Server Certificate .....	239
Renewing Certificates for the Subsystems .....	239
Step 1. Plan for Certificate Renewal .....	240
Step 2. Renew the Existing Certificate .....	241
Step 3. Install the Renewed Certificate .....	241
Step 4. Deploy the Renewed Certificate .....	242
Deploying Certificate Manager's Renewed CA Signing Certificate .....	242
Deploying Registration Manager's Renewed Signing Certificate .....	243
Deploying Data Recovery Manager's Renewed Transport Certificate ..	244
Deploying a Subsystem's Renewed SSL Server Certificate .....	246
Managing the Certificate Database .....	246
Viewing the Certificate Database Contents .....	247
Deleting a Certificate from the Certificate Database .....	249
Changing the Trust Settings of a CA Certificate .....	250
Installing a New CA Certificate in the Certificate Database .....	254
Installing a CA Certificate Chain in the Certificate Database .....	255

## Part 4 Authentication

<b>Chapter 9 Introduction to Authentication .....</b>	<b>259</b>
Privileged-User Authentication .....	260
Authentication of Administrators .....	260
Authentication of Agents .....	262



End-Entity Authentication During Certificate Enrollment .....	265
Manual Authentication .....	266
Directory-Based Authentication .....	268
Plug-in Module for User ID- and Password-Based Authentication .....	270
Configurable Parameters .....	270
Directory-Based Authentication with PINs .....	275
Plug-in Module for User ID-, Password-, and PIN-Based Authentication ..	276
Configurable Parameters .....	276
End-Entity Authentication During Certificate Renewal .....	282
End-Entity Authentication During Certificate Revocation .....	283
<b>Chapter 10 Using the PIN Generator Tool .....</b>	<b>285</b>
Locating the PIN Generator Tool .....	286
The setpin Command .....	286
Command-Line Syntax .....	286
Arguments .....	286
Example .....	291
How the Tool Works .....	292
Input File .....	295
Output File .....	296
How PINs Are Stored in the Directory .....	298
Exit Codes .....	298
Generating PINs .....	299
Step 1. Check the Directory for User Entries .....	300
Step 2. Update the Directory Schema .....	300
Updating Netscape Directory Server 3.x Schema .....	301
Updating Netscape Directory Server 4.x Schema .....	303
Step 3. Prepare the Input File .....	306
Step 4. Run the Command Without the Write Option .....	306
Step 5. Check the Output File .....	306
Step 6. Run the Command Again with the Write Option .....	307

Delivering PINs to End Entities .....	307
<b>Chapter 11 Configuring Authentication for End Entities .....</b>	<b>309</b>
Authentication Management .....	310
Authentication Management from the CMS Window .....	310
Authentication Instance Tab .....	311
Authentication Plugin Registration Tab .....	312
Authentication Parameters in the Configuration File .....	313
Authentication Plug-in Implementation and Instance .....	314
Managing Authentication Instances .....	316
Adding an Authentication Instance .....	317
Deleting an Authentication Instance .....	322
Modifying an Authentication Instance .....	322
Managing Authentication Plug-in Modules .....	325
Registering an Authentication Plug-in Module .....	325
Deleting an Authentication Plug-in Module .....	327
<b>Chapter 12 Developing Authentication Plug-ins .....</b>	<b>329</b>
Authentication Subsystem Architecture .....	330
How the Architecture Works .....	331
How Authentication Managers Are Used .....	332
Customizing Authentication .....	333
Step 1. Decide on an Authentication Scheme .....	334
Step 2. Write the Authentication Plug-in Module .....	334
Authentication Manager Plug-in API .....	334
Compiling and Installing Authentication Manager Plug-ins .....	335
Authentication Manager Examples .....	339
Step 3. Register the Authentication Manager Plug-in Module .....	340
Step 4. Create an Instance of the Authentication Plug-in Module .....	340
Step 5. Customize the End-Entity Enrollment Forms .....	340
 <b>Part 5 Job Scheduling and Notification</b>	
<b>Chapter 13 Introduction to Job Scheduling and Notifications ..</b>	<b>345</b>
Built-in Job Plug-in Modules .....	346
Certificate Renewal Notifications .....	347

Plug-in Module for Automated Renewal Notifications .....	347
Notification of Request Queue Status .....	353
Plug-in Module for Sending Notifications of Request Queue Status .....	354
Directory Update and Notification .....	358
Plug-in Module for Removing Expired Certificates from the Directory .....	359
Schedule for Executing Jobs .....	363
Event-Driven Notifications .....	364
Notifications of Certificate Issuance to End Entities .....	365
Configuring a Subsystem to Send Notifications to End Entities .....	366
Notification of New Request in Queue .....	367
Configuring a Subsystem to Send Request Queue Notifications .....	368
Customizing Notification Messages .....	370
Templates for Event-Triggered Notifications .....	370
Templates for Summary Notifications .....	372
Customizing Message Templates .....	374
Tokens Available in Message Templates .....	374
Tokens for Certificate Issuance Notifications to End Entities .....	375
Tokens for Renewal Notification Messages .....	376
Tokens for Request In Queue Notification Messages .....	378
Tokens for Directory Update Notification Messages .....	379
<b>Chapter 14 Configuring Jobs .....</b>	<b>381</b>
Job Management .....	382
Job Management from the CMS Window .....	382
Job Instance Tab .....	383
Job Plugin Registration Tab .....	384
Job Scheduler Parameters in the Configuration File .....	385
Job Plug-in Implementation and Instance .....	387
Managing Jobs .....	387
Adding a Job .....	387
Deleting a Job .....	391
Modifying a Job .....	391
Setting the Job Scheduler Frequency .....	394

Managing Job Plug-in Modules .....	396
Registering a Job Plug-in Module .....	396
Deleting a Job Scheduler Plug-in .....	398

## Part 6 Policies

<b>Chapter 15 Introduction to Policy</b> .....	401
What Is Policy? .....	402
Policy Rules .....	403
Types of Policy Rules .....	403
Using Predicates in Policy Rules .....	404
Expression Support for Predicates .....	404
Attributes for Predicates .....	406
Policy Processor .....	408
Built-in Policy Plug-in Modules .....	409
Constraints-Specific Policy Plug-in Modules .....	409
Default Revocation Policy .....	411
DSA Key Constraints Policy .....	413
Key Algorithm Constraints Policy .....	416
Renewal Validity Constraints Policy .....	419
RSA Key Constraints Policy .....	422
Validity Constraints Policy .....	426
Extension-Specific Policy Plug-in Modules .....	431
Authority Key Identifier Extension Policy .....	432
Basic Constraints Extension Policy .....	435
CRL Distribution Point Extension Policy .....	438
Key Usage Extension Policy .....	446
Netscape Certificate Type Extension Policy .....	449
Subject Alternate Name Extension Policy .....	453
Subject Key Identifier Extension Policy .....	455
<b>Chapter 16 Configuring Policies</b> .....	459
Policy Management .....	460
Policy Management from the CMS Window .....	460
Policy Rules Management Tab .....	461

Policy Plugin Registration Tab .....	462
Policy Parameters in the Configuration File .....	463
Policy Plug-in Implementation and Rule .....	466
Managing Policy Rules .....	468
Adding a Policy Rule .....	468
Deleting a Policy Rule .....	472
Modifying a Policy Rule .....	473
Reordering Policy Rules .....	477
Managing Policy Plug-in Modules .....	479
Registering a Policy Plug-in Module .....	479
Deleting a Policy Plug-in Module .....	481

## **Part 7 LDAP Publishing**

<b>Chapter 17 Introduction to LDAP Publishing .....</b>	<b>485</b>
What Is LDAP Publishing? .....	486
Timing of Directory Updates .....	488
Objects Published by the Certificate Manager .....	489
Objects Published by the Registration Manager .....	490
Directory Update Process .....	490
Object-Mapping Rules .....	491
Built-in Mapper Classes .....	491
How Mapping by DN Components Works .....	496
Object-Publishing Rules .....	498
Built-in Publisher Classes .....	498
Directory Schema Requirements .....	499
Required Schema for Publishing End-Entity Certificates .....	500
Required Schema for Publishing CA Certificates .....	500
Required Schema for Publishing CRLs .....	501
Directory Synchronization .....	501
<b>Chapter 18 Configuring Subsystems for LDAP Publishing .....</b>	<b>503</b>
Setting Up the Directory for Publishing .....	504
Step 1. Verify the Directory Schema .....	504
Step 2. Add an Entry for the CA .....	504

Step 3. Identify an Entry That Has Write Access .....	505
Step 4. Add Entries for End Entities .....	507
Configuring a Certificate Manager for LDAP Publishing .....	507
Identifying a Certificate Manager's Publishing Directory .....	508
Configuring Mapper and Publisher Classes for the CA Certificate .....	511
Configuring Mapper and Publisher Classes for End-Entity Certificates .....	513
Configuring a Registration Manager for LDAP Publishing .....	515
Identifying a Registration Manager's Publishing Directory .....	515
Configuring Mapper and Publisher Classes for End-Entity Certificates .....	518
Manually Updating Certificate Information in the Directory .....	520
<b>Chapter 19 Publishing CRLs .....</b>	<b>523</b>
CRL Authorities .....	524
CRL Issuing Points .....	524
Reasons for Revoking a Certificate .....	525
Updating CRLs Automatically .....	526
Configuring a Certificate Manager for Publishing CRLs .....	527
Updating CRLs Manually .....	529

## **Part 8 Agent and End-Entity Interfaces**

<b>Chapter 20 Introduction to End-Entity and Agent Interfaces ....</b>	<b>533</b>
End-Entity Services .....	533
How Client Type Determines the End-Entity Interface .....	535
Certificate Request Formats Specific to End Entities .....	536
Configuring End-Entity Interaction with Subsystems .....	537
Enabling End-Entity Interaction with a Certificate Manager .....	538
Enabling End-Entity Interaction with a Registration Manager .....	539
Agent Services .....	541
Certificate Manager Agent Services .....	541
Registration Manager Agent Services .....	542
Data Recovery Manager Agent Services .....	543
Accessing the Agent Services Interface .....	544

<b>Chapter 21 Customizing End-Entity and Agent Interfaces</b>	547
What You Need to Know	548
HTTP, Query URLs, and HTML Forms	548
JavaScript	548
How the Forms Work	549
Requests Sent to the Server	549
Responses and Output Templates	550
Errors and the Error Template	553
Summary of End-Entity Forms and Templates	553
Locating End-Entity Forms and Templates	555
Forms for Certificate Enrollment	555
Forms for Certificate Renewal	557
Forms for Certificate Revocation	557
Forms for Certificate Retrieval	558
Forms for Key Recovery	559
Other Forms	560
Output Templates for End-Entity Operations	561
Summary of Agent Forms and Templates	563
Structure of the Agent Services Interface	563
Locating Agent Forms and Templates	564

## Part 9 Logs

<b>Chapter 22 Introduction to Logs</b>	567
Logs Maintained by Certificate Management System	568
Services That Are Logged	569
Log Levels (Message Categories)	570
Log File Locations	572
Log File Naming Conventions	573
Active Log File Naming Convention	573
Rotated Log File Naming Convention	573
Buffered Versus Unbuffered Logging	574
Rotation of Log Files	574
Timing of Log File Rotation	575
Location of Rotated Log Files	575

Deletion of Log Files .....	575
How to Conserve Disk Space .....	575
Timing of Log File Deletion .....	576
Archiving of Rotated Log Files .....	576
<b>Chapter 23 Managing Logs .....</b>	<b>577</b>
Management of Logs .....	578
Log Management from the CMS Window .....	578
Log Parameters in the Configuration File .....	580
Configuring Logs .....	581
Configuring System Logs .....	581
Configuring Error Logs .....	583
Configuring Audit Logs .....	585
Monitoring Logs .....	586
Monitoring System Logs .....	587
Monitoring Error Logs .....	589
Monitoring Audit Logs .....	592
Using System Tools for Monitoring the Server (Windows NT Only) .....	594
Logging to Windows NT Event Log .....	594
Using Event Viewer .....	596
Signing Log Files .....	597

## **Part 10 Issuance and Management of End-Entity Certificates**

<b>Chapter 24 Issuing and Managing End-Entity Certificates .....</b>	<b>603</b>
Certificate Issuance to Servers .....	603
How the Manual Server Enrollment Process Works .....	604
Getting Server SSL Certificates for Netscape Servers .....	606
Getting Certificates for Version 3.x Servers .....	607
Getting Certificates for Netscape Version 4.x Servers .....	612
Certificate Issuance to Routers .....	613
Step 1. Find the Required Information .....	613
Step 2. Generate the Key Pair for the Router .....	615
Step 3. Request the CA's Certificate .....	615
Step 4. Submit the Certificate Request to the CA .....	616



Example .....	617
Certificate Renewal .....	619
Renewal of Client Certificates .....	619
Renewal of Server Certificates .....	621
Certificate Revocation .....	621
<b>Chapter 25 Recovering Encrypted Data .....</b>	<b>623</b>
PKI Setup for Key Archival and Recovery .....	624
Clients That Can Generate Dual Key Pairs .....	624
Data Recovery Manager .....	625
Forms for Users and Key Recovery Agents .....	625
Key Archival Process .....	626
Why You Should Archive Keys .....	626
Where the Keys are Stored .....	626
How Key Archival Works .....	627
Key Recovery Process .....	629
Key Recovery Agents and Their Passwords .....	630
Secret Sharing of Storage Key Password .....	630
Interface for the Key Recovery Process .....	631
Local Versus Remote Key Recovery Authorization .....	632
How Agent-Initiated Key Recovery Works .....	633
Key Recovery Agent Scheme .....	636
Changing the Key Recovery Agent Scheme .....	636
Changing Key Recovery Agents' Passwords .....	639
Setting Up Key Archival and Recovery Process .....	641
Setting Up the Key Archival Process .....	641
Step 1. Deploy Clients That Can Generate Dual Key Pairs .....	641
Step 2. Connect the Enrollment Authority and the Data Recovery Manager .....	642
Step 3. Customize the Certificate Enrollment Form .....	643
Step 4. Configure Key Archival Policies .....	650
Step 5. Test Your Key Archival Setup .....	650
Setting Up the Key Recovery Process .....	651
Step 1. Verify the m of n scheme .....	652

Step 2. Facilitate the Key Recovery Agents to Change the Passwords .	652
Step 3. Determine the Authorization Mode for Key Recovery .....	653
Step 4. Customize the Key Recovery Form .....	653
Step 5. Configure Key Recovery Policies .....	653
Step 6. Test Your Key Recovery Setup .....	654

## Part II Appendixes

<b>Appendix A Distinguished Names</b> .....	659
What Is a Distinguished Name? .....	659
Distinguished Name Components .....	660
Root Distinguished Name .....	661
Base Distinguished Name .....	661
Role of Distinguished Names in Certificates .....	662
DNs in End-Entity Certificates .....	663
DNs in CA Certificates .....	664
Selecting DN for Certificates .....	664
<b>Appendix B Backing Up and Restoring Data</b> .....	665
Before Backing Up and Restoring Data .....	665
What Is a Backup? .....	666
Why You Should Back Up Data .....	666
Guidelines for Creating a Backup .....	666
What Is a Restore? .....	667
When to Restore Data .....	667
Guidelines for Restoring Data .....	667
Backing Up the CMS Configuration and Data .....	668
Step 1. Back Up the Configuration Files .....	668
Step 2. Back Up the Key Pairs .....	669
Step 3. Back Up the Internal Database .....	669
Restoring the CMS Configuration and Data .....	672
<b>Appendix C Command-Line Utilities</b> .....	673
Summary of Command-Line Utilities .....	673
Location of Command-Line Utilities .....	676

ASCII to Binary Tool .....	676
Availability .....	676
Syntax .....	677
Example .....	677
Binary to ASCII Tool .....	677
Availability .....	677
Syntax .....	677
Example .....	678
Pretty Print Certificate Tool .....	678
Availability .....	678
Syntax .....	678
Example .....	679
Pretty Print CRL Tool .....	681
Availability .....	681
Syntax .....	681
Example .....	681
dumpasn1 Tool .....	683
<b>Appendix D Certificate Database Tool .....</b>	<b>685</b>
Availability .....	686
Syntax .....	686
Options and Arguments .....	686
Usage .....	694
Examples .....	695
Creating a New Certificate Database .....	695
Listing Certificates in a Database .....	695
Creating a Certificate Request .....	696
Creating a Certificate .....	697
Adding a Certificate to the Database .....	697
Validating a Certificate .....	699
<b>Appendix E Key Database Tool .....</b>	<b>701</b>
Availability .....	702
Syntax .....	702
Options and Arguments .....	702

Usage .....	705
Examples .....	706
Creating a Key Database .....	706
Generating a New Key .....	707
Displaying Public Key Information .....	708
Listing Key IDs .....	708
Deleting a Private Key .....	709
<b>Appendix F Netscape Signing Tool .....</b>	<b>711</b>
Introduction to Netscape Signing Tool .....	712
What Is Netscape Signing Tool? .....	712
JAR Format and JAR Archives .....	713
What Signing a File Means .....	714
Object-Signing Certificates .....	715
Using Netscape Signing Tool .....	716
Getting Ready to Use Netscape Signing Tool .....	716
Setting Up Your Certificate .....	717
Listing Available Certificates .....	718
Signing a File .....	719
Using Netscape Signing Tool with a ZIP Utility .....	720
Tips and Techniques .....	720
SignTool Syntax and Options .....	722
Command Syntax .....	722
Command Options .....	723
Command File Syntax .....	730
Command File Keywords and Example .....	731
Generating Test Object-Signing Certificates .....	733
Generating the Keys and Certificate .....	733
Using Netscape Signing Tool with Smart Cards .....	735
What Is a Smart Card? .....	735
Setting Up a Smart Card .....	736
Using the -M Option to List Smart Cards .....	737
Using Netscape Signing Tool and a Smart Card to Sign Files .....	738

Netscape Signing Tool and FIPS-140-1 .....	738
Using FIPS-140 Mode .....	739
Verifying FIPS Mode .....	739
Answers to Common Questions .....	740
<b>Appendix G SSL Strength Tool .....</b>	<b>745</b>
Availability .....	745
Syntax .....	746
Options and Arguments .....	746
Usage .....	747
Restricting Ciphers .....	748
Export Policy and Step-up .....	748
Examples .....	748
Example 1 .....	749
Example 2 .....	749
Example 3 .....	750
<b>Appendix H SSL Debugging Tool .....</b>	<b>751</b>
Availability .....	751
Description .....	752
Syntax .....	752
Options .....	752
Examples .....	754
Example 1 .....	755
Command .....	755
Output .....	755
Example 2 .....	758
Command .....	758
Output .....	758
Example 3 .....	761
Command .....	761
Output .....	761
Example 4 .....	762
Command .....	763
Output .....	763

Usage Tips .....	764
<b>Index</b> .....	765

# About This Guide

The *Administrator's Guide* explains how to administer Netscape Certificate Management System (CMS). It assumes that you have read the installation documentation and have successfully installed Certificate Management System.

This preface has the following sections:

- What's in This Guide (page 23)
- Who Should Read This Guide (page 23)
- What You Should Already Know (page 24)
- Conventions Used in This Guide (page 25)
- Where to Go for Related Information (page 27)

## What's in This Guide

This guide explains how to configure, customize, and maintain Certificate Management System, and use it for issuing and managing certificates to various end entities, such as clients (users), servers, VPN clients, and Cisco™ routers.

## Who Should Read This Guide

This guide is intended for Certificate Management System administrators.

# What You Should Already Know

This guide assumes that you

- Are familiar with the basic concepts of public-key cryptography and the Secure Sockets Layer (SSL) protocol.
  - SSL cipher suites
  - The purpose of and major steps in the SSL handshake
- Understand the concepts of intranet, extranet, and the Internet security and the role of digital certificates in a secure enterprise. These include the following topics:
  - Encryption and decryption
  - Public keys, private keys, and symmetric keys
  - Significance of key lengths
  - Digital signatures
  - Digital certificates, including various types of digital certificates
  - Their role of digital certificates in a public-key infrastructure (PKI)
  - Certificate hierarchies

If you are new to these concepts, we recommend you read the security-related documents available online at this URL:

<http://developer.netscape.com/docs/manuals/index.html?content=security.html>

You may also refer to the security-related appendixes (D and E) of the accompanying manual, *Managing Servers with Netscape Console*.

- Are familiar with the role of Netscape Console in managing Netscape version 4.x servers. Otherwise, see the accompanying manual, *Managing Servers with Netscape Console*.
- Have read the *Netscape Certificate Management System Installation and Deployment Guide*.



# Conventions Used in This Guide

The following conventions are used in this guide:

- `Monospaced font`

This typeface is used for any text that appears on the computer screen or text that you should type. It's also used for filenames, functions, and examples.

Example: `Server Root` is the directory where the CMS binaries are kept.

- *Italic*

Italic type is used for emphasis, book titles, and glossary terms.

Example: This control depends on the access permissions the *superadministrator* has set up for you.

- Text within “quotation marks”

Cross-references to other topics within this guide.

Example: For more information, see “Issuing a Certificate to a New User” on page 154.

- **Boldface**

Boldface type is used for various UI components such as captions and field names, and the terminology explained in the glossary.

Example:

**Rotation frequency.** From the drop-down list, select the interval at which the server should rotate the active error log file. The available choices are Hourly, Daily, Weekly, Monthly, and Yearly. The default selection is Monthly.

- Monospaced [ ]

Square brackets enclose commands that are optional.

Example:

```
PrettyPrintCert <input_file> [<output_file>]
```

<input\_file> specifies the path to the file that contains the base-64 encoded certificate.

<output\_file> specifies the path to the file to write the certificate. This argument is optional; if you don't specify an output file, the certificate information is written to the standard output.

- Monospaced <>

Angle brackets enclose variables. When following examples, replace the angle brackets and their text with text that applies to your situation. For example, when path names appear in angle brackets, substitute the path names used on your computer.

Example: Using Netscape Communicator 4.04 or later, enter the URL for the administration server:

```
http://<server_name>.<your_domain>.<domain>:<port_number>
```

- /

A slash is used to separate directories in a path. If you use the Windows NT operating system, you should replace / with \ in paths.

Example: Except for the Security Module Database Tool, you can find all the other command-line utilities at this location:

```
<server_root>/bin/cert/tools/...
```

- Sidebar text

Sidebar text marks important information. Make sure you read the information before continuing with a task.

Examples:

**Note** You can use Netscape Console only when Administration Server is up and running.

**Caution** A caution note documents a potential risk of losing data, damaging software or hardware, or otherwise disrupting system performance.

**Unix** Marks text that applies only to the Unix versions of Certificate Management System.

**Windows NT** Marks text that applies only to the Windows NT version of Certificate Management System.

## Where to Go for Related Information

This section summarizes the documentation that ships with Netscape Certificate Management System, using these conventions:

- `<server_root>` is the directory where the CMS binaries are kept (specified during installation).
- `<instance_id>` is the ID for this instance of Netscape Certificate Management System (specified during installation).

The documentation set for Netscape Certificate Management System includes the following:

- *Managing Servers with Netscape Console* provides background information on basic cryptography concepts and the role of Netscape Console.
  - For the HTML version, see `<server_root>/manual/en/admin/help/contents.htm`
- *Netscape Certificate Management System Installation and Deployment Guide* describes how to plan for and install Netscape Certificate Management System. To access the installation and configuration information from within the CMS Installation Wizard, click any help button.
  - The HTML version of this guide is located at `<server_root>/manual/en/cert/dep_gide/contents.htm`.
  - The PDF version of this guide is located at `<server_root>/manual/en/cert/pdf/cs40_dep.pdf`.

- *Netscape Certificate Management System Administrator's Guide* (this guide) provides detailed reference information on CMS administration interfaces. To access this information from the CMS window within Netscape Console, click any help button.
  - The HTML version of this guide is located at `<server_root>/manual/en/cert/adm_gide/contents.htm`.
  - The PDF version of this guide is located at `<server_root>/manual/en/cert/pdf/cs40_adm.pdf`.
- *Netscape Certificate Management System Agent's Guide* provides detailed reference information on CMS agent interfaces. To access this information from the Agent Services pages, click any help button.
  - The HTML version of this guide is located in `<server_root>/<instance_id>/web/agent/manual/agt_gide/contents.htm`.
  - The PDF version of this guide is located at `<server_root>/manual/en/cert/pdf/cs40_agt.pdf`.
- End-entity help (online only, not printed) provides detailed reference information on CMS end-entity interfaces. To access this information from the end-entity pages, click any help button.
  - The HTML version of this guide is located at `<server_root>/<instance_id>/web/ee/manual/ee_gide/contents.htm`.

For a complete list of all documentation that ships with Netscape Certificate Management System, including documentation for Directory Server, see Documentation Summary, located at `<server_root>/manual/index.html`.

For the latest information about Netscape Certificate Management System, including current release notes, technical notes, and deployment information, see <http://home.netscape.com/eng/server/cms/>.

**Important** Do not change the default location of any of the HTML files; they are used for online help. You may move the PDF files to another location.

# 1

## *Netscape Certificate Management System*

*Chapter 1*    Introduction to Certificate Management System

*Chapter 2*    Administration Tasks and Tool

*Chapter 3*    Configuration



# Introduction to Certificate Management System

This chapter introduces Netscape Certificate Management System (CMS), a highly configurable set of software components and tools for creating, deploying, and managing certificates. Based on open standards for certificate management, Certificate Management System leverages Netscape Directory Server and Netscape Console to provide a complete, scalable, high-performance certificate management solution for extranets and intranets.

Whether you are looking for a security solution for your enterprise or setting up an independent certificate authority (CA) service, Certificate Management System offers a robust, customizable, and scalable foundation for your public-key infrastructure (PKI).

The chapter has the following sections:

- Overview (page 32)
- Key Features (page 33)
- System Architecture (page 39)
- Entry Points for Various Types of Users (page 45)

# Overview

Certificate Management System provides a highly scalable, easily deployable certificate infrastructure for supporting encryption, authentication, tamper detection, and digital signatures in networked communications. It is based on open standards and protocols such as Public-Key Cryptography Standard (PKCS) #7, 10, 11, and 12, Secure Sockets Layer (SSL), Lightweight Directory Access Protocol (LDAP), and the X.509 certificate formats recommended by the International Telecommunications Union (ITU). Certificate Management System is highly customizable and configurable, permitting rapid integration with existing client and server software, customer databases, security systems, and authentication procedures.

You can use Certificate Management System to set up and manage your own public-key infrastructure or to deploy a public certification authority. Certificate Management System meets the needs of an enterprise, leveraging your existing enterprise resources and services, and will grow with your business needs to meet the demand of Internet-scale deployments.

With Certificate Management System, you can do the following operations:

- Process certificate requests from various end entities, such as clients, servers, routers, and virtual private network (VPN) clients, and issue certificates that conform to X.509 version 1 or version 3 standards.
- Employ specific authentication mechanisms for end-entity certificate enrollment, renewal, and revocation.
- Specify policy restrictions on certificate-related operations, such as certificate formulation, issuance, renewal, and revocation.
- Specify policy restrictions on key-related operations, such as archival and recovery of end users' encryption private keys.
- Revoke certificates, and maintain and publish a list of revoked certificates.
- Search for certificates issued by the server.
- Set up hierarchies of certificate authorities—multiple subordinate CAs chained up to a root CA.
- Publish certificate information and the list of revoked certificates to an LDAP-compliant directory, such as Netscape Directory Server, and maintain this information.



# Key Features

Certificate Management System has many core features:

## Support for open standards

With its support for open standards, Certificate Management System gives organizations confidence that they will be able to communicate within a heterogeneous computing environment. Specifically, Certificate Management System does the following:

- Formulates, signs, and issues industry-standard X.509 version 1 and 3 public-key certificates; version 3 certificates include extensions that make it easy to include organization-defined attributes. This means that you can use these certificates for extranet and Internet authentication as well.
- Supports RSA public-key algorithm for signing and encryption, DSA public-key algorithm for signing, and MD5 and SHA-1 for hashing.
- Supports signature and encryption key lengths of up to 2048 bits for domestic use and 512 bits for export use.
- Supports multiple formats for certificate requests using related standards, such as HTML (KEYGEN/SPAC), PKCS#10, CMMF, CEP, and HTTP.
- Supports certificate formats that encompass certificates for SSL-based client and server authentication, secure Multipurpose Internet Mail Extensions (S/MIME) message signing and encryption, object signing, VPN clients, and Cisco™ routers. The server can also issue financial certificates for Netscape and Microsoft™ servers with step-up functionality for export browsers (with appropriate U.S. Department of Commerce approval).
- Supports PKCS #12, an industry-standard certificate and key export-import format that encrypts key and certificate information with a user password or passphrase. This format bundles certificates and private keys with passwords and publishes them to LDAP directories for users to download into their client environment.
- Publishes certificates and certificate revocation lists (CRLs) to any LDAP-compliant directory over LDAP and HTTP/HTTPS connections.
- Supports generation and publication of CRLs conforming to X.509 version 1 and 2.

## Separate subsystems for certificate and key operations

Certificate Management System includes three servers, the *Certificate Manager*, *Registration Manager*, and *Data Recovery Manager*.

- The Certificate Manager functions as the certificate authority (CA); it is the entity named in the issuer field of a certificate. The Certificate Manager can sign and revoke certificates and generate CRLs. It can accept certificate requests directly from end entities and via Registration Managers to which it has delegated certain certificate management functions, such as authentication of an end entity. The Certificate Manager also maintains a database of issued certificates so that it can track renewal, expiration, and revocation.
- The Registration Manager is an optional component in the PKI; it is a subordinate server to which a Certificate Manager can delegate some certificate management functions. For example, a Registration Manager may perform tasks, such as end-entity authentication and formulation of the certificate request for the Certificate Manager.
- The Data Recovery Manager is an optional component in the PKI. It provides key archival and recovery services for end users' encryption private keys.

## Single CA supports multiple registration authorities

Certificate Management System lets you separate the registration process from the certificate-signing process with the help of Registration Managers. You can run multiple Registration Managers remotely, all reporting to a single CA—a Certificate Manager—to verify user identities and process certificate signing requests. The remote Registration Managers forward their completed and approved requests to the Certificate Manager for it to sign and issue the certificate automatically.

The certificate requests submitted by the remote Registration Managers are standards-based, so that you can integrate your own registration process into the certificate management process.

The Certificate Manager's ability to support multiple Registration Managers makes it more scalable and also adds an extra layer of security for the CA. For example, you can set a policy that requires all clients to go through a remote Registration Manager, and then have the remote Registration Manager route all client requests to the Certificate Manager located inside a firewall.

## **Ability to function as both a root and a subordinate CA in a CA hierarchy**

Certificate Management System can function as a root (or parent) CA (in which case the server signs its own CA signing key as well as other CA signing keys) enabling you to create your own CA hierarchy. You can also install the server to function as a subordinate CA (in which case the server gets its CA signing key signed by another CA) in an existing CA hierarchy.

## **Ability to function as a linked CA**

Certificate Management System can function as a *linked CA*, chaining up to many third-party or public CAs for validation; this provides cross-company trust, so applications can verify certificate chains outside the company certificate hierarchy.

## **PKCS #11 hardware support for smart cards and crypto accelerators**

Certificate Management System supports smart cards and crypto accelerators provided by various third-party vendors of PKCS #11 version 2.1-compliant products. For a complete list of vendors, see the information available at this URL: <http://home.netscape.com/cms/v4.0/index.html>

You can configure the server to use different PKCS #11 modules to generate and store key pairs (and certificates) for the Certificate Manager, Registration Manager, and Data Recovery Manager. Using hardware for key storage (especially for Certificate Manager and Data Recovery Manager key pairs) reduces the risk of key compromise, because hardware tokens don't reveal keys or provide means for them to be revealed, once the keys are generated in the hardware.

## **Support for Netscape client and server products; client independence for non-Netscape products**

Certificates issued by Certificate Management System work with existing Netscape client and server products that support SSL. The certificates also work (out of the box) with a variety of non-Netscape, standards-compliant applications. For a complete list of these products, see the information available at this URL: <http://home.netscape.com/cms/v4.0/index.html>

## **Highly scalable certificate data store**

Certificate Management System uses a highly scalable, high-performance certificate storage facility—a built-in, preconfigured version of Netscape Directory Server 4.x—enabling you to issue and manage a large number of certificates.

## **Flexible end-entity registration services framework**

The registration services framework for end entities includes the most commonly expected PKI features: manual, directory-based, and directory- plus PIN-based enrollment; certificate-authenticated renewals and revocations (based on SSL client authentication); certificate life-cycle operations that include automated certificate renewal and expiration notifications. These features are available out of the box for both Certificate Manager and Registration Manager.

## **Built-in plug-in modules for authentication, policy, job scheduling, and LDAP publishing**

Certificate Management System simplifies the details involved in certificate issuance and management with its built-in, configurable, and extensible authentication, policy, job scheduling, and LDAP publishing components. Each of these components come with customizable plug-in modules. For example, you can configure policy modules to determine the outcome of operations, such as certificate formulation (extensions, signing algorithm, key length, validity period, and so on), issuance, renewal, and revocation.

## **Single administration point achieved via LDAP-compliant directory integration**

Certificate Management System works seamlessly with any LDAP-compliant directory services for easy distribution of certificates and CRLs, thus lowering the cost of information management. The shared directory architecture enables you to manage users, including their security credentials and other shared data, at a single place. Certificate Management System can do the following:

- Authenticate users based on the information that exists in the LDAP directory.
- Integrate certificate-related information with the user and group information that exists in the LDAP directory.

- Automatically publish certificates (when they are issued) and CRLs (when created or on a periodic basis) to the LDAP directory, from which they can be easily distributed to clients and servers.
- Automatically delete expired and revoked certificates from the directory.
- Connect to the directory using password-based (basic), certificate-based (strong) authentication, or both.

### **Supports certificate generation for dual key pairs—separate key pairs for signing and encrypting mail messages**

To support separate key pairs for signing and encrypting data, Certificate Management System supports generation of dual certificates for end entities capable of generating dual key pairs. If a client makes a certificate request for dual key pairs, the server issues two separate certificates.

### **Key archival and recovery for encryption private keys**

If your organization uses S/MIME to encrypt mail messages, you can use the key archival feature offered by Certificate Management System to back up users' encryption private keys. This feature is useful when a key becomes unavailable—as, for instance, in the following cases:

- An employee loses an encryption private key (for example after a disk crash or by forgetting the password to the key file) and is unable to read previously encrypted data.
- An employee leaves the company, and company officials need to perform an audit that requires gaining access to the employee's encrypted data.

### **Encrypted key storage and password-protected recovery**

Certificate Management System stores users' encryption private keys in an encrypted key repository. Keys can be retrieved only by authorized key recovery agents. The key repository is encrypted using a Data Recovery Manager's storage private key, which is protected with one or more recovery agents' passwords. Only these designated recovery agents can authorize and initiate a key recovery process.

## **Extensive audit and log records for detection of tampering**

Certificate Management System maintains audit trails for all events—certificate requests and issuance, revocation requests, CRL publication, and so on. These audit records enable you to detect any unauthorized access or activity. In addition, extensive system and error logs record various events and system errors so that you can monitor and debug the system. All log records are stored in your local file system for quick and easy retrieval.

## **Supports signing of log files for tamper detection**

Certificate Management System allows you to sign log files digitally before archiving them or distributing them for audit purposes. This feature enables you to check whether the log files were tampered with after being signed.

## **Java SDK extension mechanism for customization**

The Java-based software development kit (SDK) provided with Certificate Management System includes APIs for customizing different aspects of the system. You can write the following custom modules:

- Authentication—for authenticating end entities during certificate enrollment.
- Policy—for setting the rules applied by the individual subsystems.
- Jobs—for PKI-related jobs that run with the individual systems.
- Mapper and publisher classes—for publishing certificates and CRLs to an LDAP-compliant directory.
- Gateway for end-entity and agent forms or HTTP interfaces—for customizing end-entity and agent interfaces.

## **Easy migration path from Netscape Certificate Server 1.0x**

Certificate Management System provides an easy migration path from Netscape Certificate Server 1.0x. The server includes a command-line-based migration tool that extracts the contents of a Certificate Server 1.0x database, including keys and certificates, and puts them in three platform-independent files. You then import the contents of these files into the internal database of Certificate Management System.

## Easy, GUI-based server installation and management

An installation wizard automates the installation and initial configuration process, helping you install Certificate Management System quickly and easily. Then after installation, you can locally or remotely administer Certificate Management System from various computers on your network (using the encryption, message integrity, and authentication services of SSL) with the help of an administration interface called the Certificate Management System window or the CMS window.

# System Architecture

Certificate Management System comprises three servers, or *main subsystems*, and a number of system-level components that are shared by these servers. The main subsystems are:

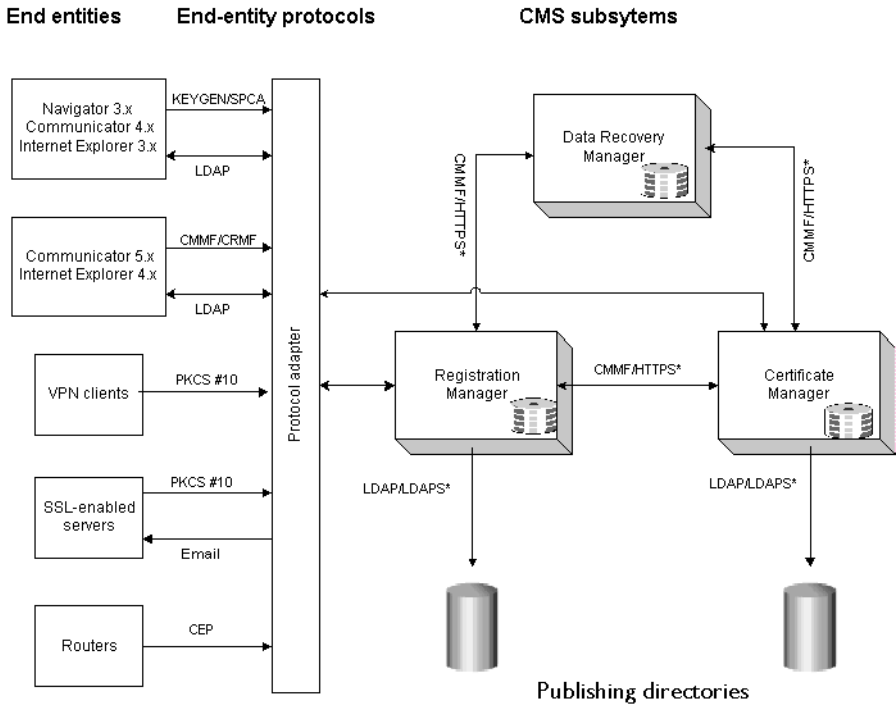
- Certificate Manager
- Registration Manager
- Data Recovery Manager

The system-level components (also referred to as *subsystems*) are:

- Authentication
- Policy
- Job scheduling and notification
- Logging
- LDAP publishing
- Internal database

Figure 1.1 illustrates the high-level architecture of Certificate Management System.

Figure 1.1 Certificate Management System architecture



\* HTTPS or LDAPS with SSL client authentication

Table 1.1 gives an overview of each component's tasks. For detailed explanation for various components and how they interact, see the *Netscape Certificate Management System Installation and Deployment Guide*.



Table I.I Certificate Management System components

Component/Subsystem	Description
End entities	<p>Various end entities that can request certificates from Certificate Management System:</p> <ul style="list-style-type: none"> <li>• Clients, such as Netscape Navigator version 3.x; Netscape Communicator versions 4.x and later; Microsoft Internet Explorer versions 3.x, 4.x, and 5.x</li> <li>• SSL-enabled servers, such as Netscape Administration, Directory, and Enterprise Servers</li> <li>• Routers, such as Cisco routers</li> <li>• Virtual private network clients, such as Aventail™, KyberPass™, and RedCreek™</li> </ul> <p>For details on supported end-entity protocols, see “How Client Type Determines the End-Entity Interface” on page 535.</p>
End-entity protocols	<p>Various protocols that Certificate Management System supports for allowing end-entity interaction with the server. For details, see “Certificate Request Formats Specific to End Entities” on page 536.</p>
Certificate Manager	<p>Functions as the certificate authority (CA); it is the entity named in the issuer field of a certificate.</p> <p>Exposes a number of interfaces used by protocol adapters and other CMS subsystems or components to perform the fundamental tasks of certificate management; it is a service that signs and revokes certificates and generates CRLs.</p> <p>It can accept certificate requests directly from end entities as well as from Registration Managers to which it has delegated certain certificate management functions, such as authentication of an end entity.</p>

Table 1.1 Certificate Management System components (Continued)

Component/Subsystem	Description
Registration Manager	Functions as a full-fledged, remote registration front end to a Certificate Manager, enforcing policies (defined by the Policy engine) on certificate issuance, renewal, and revocation requests, key updates and recovery, and allied functions. Multiple Registration Managers can report to a single Certificate Manager (CA).
Data Recovery Manager	Handles encryption private key operations, such as key archival and recovery.
Internal database	An LDAP-compliant persistent storage system built into Certificate Management System. This database is a preconfigured version of Netscape Directory Server (version 4.x) installed transparently at the time of CMS installation.
Publishing directories	Used for publishing certificates and CRLs. Certificate Management System can publish to any LDAP-compliant directory (such as Netscape Directory Server 1.x, 3.x, and 4.x) used by organizations to maintain corporate data in a single place.

## LDAP Directory Integration

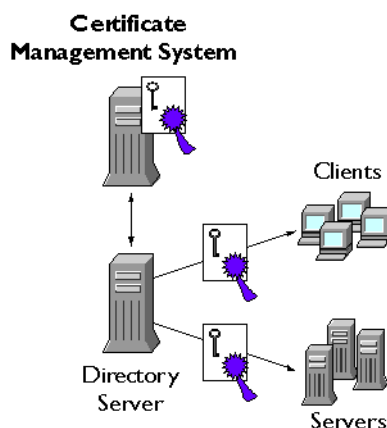
Certificate Management System can function very closely with an LDAP-compliant directory, such as Netscape Directory Server, that organizations typically use to maintain corporatwide data about user and group accounts and other network resources. You can set up Certificate Management System to automatically publish certificate information and CRLs to a directory. The advantage of publishing certificates and CRLs to the directory is multifold:

- You can keep users' certificate-related information with the rest of the user information. This way, when you update the user information, the certificate-related information automatically gets updated. For example, when you delete a user entry, the security credentials of that user automatically gets deleted from the directory.

- If you are using S/MIME-enabled clients (for example, Netscape Communicator), publishing all certificates to a central directory enables your users to import others' certificates from the global directory.

For more information on setting up Certificate Management System to publish certificates and CRLs, see “LDAP Publishing” on page 483.

Figure 1.2 Seamless integration with any LDAP-compliant directory



Seamless integration with any LDAP-compliant directory (see Figure 1.2.) makes possible the following:

- Corporate IS organizations can generate and manage certificates as an integral part of their user and group management.
- Independent CAs can issue and manage certificates to their users listed in any LDAP-compliant directory.

## How the Main Subsystems Function

You can install the Certificate Manager and Data Recovery Manager together in the same instance or separately in two different instances. Installation in separate instances requires additional configuration to connect the two subsystems together.

- If you installed the subsystems separately, then you need to know whether they are in the same server group or separate server groups.
  - If they are in the same server group, then they use the same instance of Netscape Administration Server installed for that group. This server enables you to launch the GUI-based administrative interface, Netscape Console, for both the instances. The login URL for Netscape Console is the same for both the instances.
  - If they are under separate server groups, then they use separate instances of Netscape Administration Server installed for those groups. The login URL for Netscape Console will be different for the subsystems.
- If you installed the subsystems together, then they share a common administrative interface, the CMS window, which can be launched within Netscape Console. Both the subsystems use the same internal database for storing and retrieving persistent objects such as certificates and keys. In addition, the subsystems also share various supporting services, such as the template manager and logging.

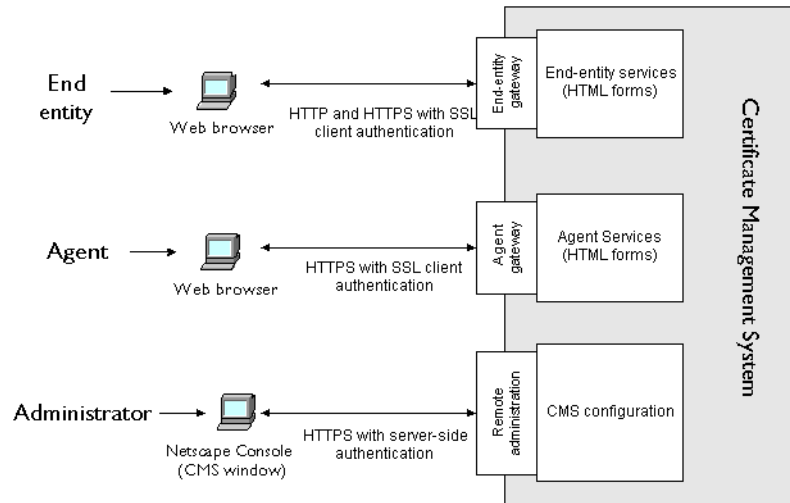
For information on Netscape Console and the CMS window, see “Administration Tasks and Tool” on page 47.

Optionally, you can also install Registration Managers as standalone entities, and connect them to a Certificate Manager or Data Recovery Manager. A Registration Manager connected to a Certificate Manager functions as a trusted front end to the Certificate Manager by receiving end-entity requests, authenticating them, and forwarding them to the Certificate Manager for signing. After receiving a response from the Certificate Manager, the Registration Manager notifies the end entity of the results. Similarly, a Certificate Manager or Registration Manager connected to a Data Recovery Manager handles end users' requests related to the archival of encryption private keys. For more information on connecting subsystems, see “Trusted Managers” on page 141.

# Entry Points for Various Types of Users

Certificate Management System provides entry points for various kinds of user interaction.

Figure 1.3 Entry points for different types of users



As illustrated in Figure 1.3, the server provides three separate user entry points; each entry point addresses the needs of a specific user type. This is explained in Table 1.2.

Table 1.2 Certificate Server user-entry points

User	Component/ Tool	CMS interface	Description
End entity	Web browser	End-entity gateway	This gateway provides the general front end for end-entity interactions with the server. Through this gateway, the Certificate Manager or Registration Manager serves the appropriate HTML forms for end-entity operations (the Data Recovery Manager does not have an end-entity interface). These include forms for certificate enrollment, retrieval, query, renewal, import, and revocation. These forms are collectively referred to as the <i>end-entity services</i> interface.
Agent	Web browser	Agent gateway	This gateway provides the general front end for agent interactions with the server. Through this gateway, a Certificate Manager, Registration Manager, or Data Recovery Manager serves the appropriate HTML forms for agent tasks. These forms are collectively referred to as the <i>Agent Services</i> interface. Accessing Agent Services is a privileged operation; agents must use designated certificates for SSL client authentication to Certificate Management System.
Administrator	Netscape Console (CMS window)	Remote administration	The remote administration interface supports a GUI-based administration tool called Netscape Console that provides the general administration and management interface for Certificate Management System. Administrators can use this tool to perform day-to-day operational and managerial duties, such as changing the server configuration, stopping and restarting the server, requesting and installing certificates, managing resources (certificates and requests), and setting up privileged-user information and associated access controls. The CMS window can only be launched from within Netscape Console. Accessing this window is a privileged operation requiring a password-based authentication to Certificate Management System.

# Administration Tasks and Tool

In administering Netscape Certificate Management System (CMS), you perform server-specific tasks such as starting, stopping, and restarting the server; changing configuration; configuring certificate issuance and management policies; adding or modifying privileged-user and group information; setting up authentication mechanisms for users who may request services from the server; performing routine server maintenance tasks; monitoring logs; and backing up server data.

To enable system administrators to accomplish these server-specific tasks quickly and easily, Certificate Management System provides a GUI-based administration tool, called the CMS window, within Netscape Console. This chapter provides an overview of both Netscape Console and the CMS window.

The chapter has the following sections:

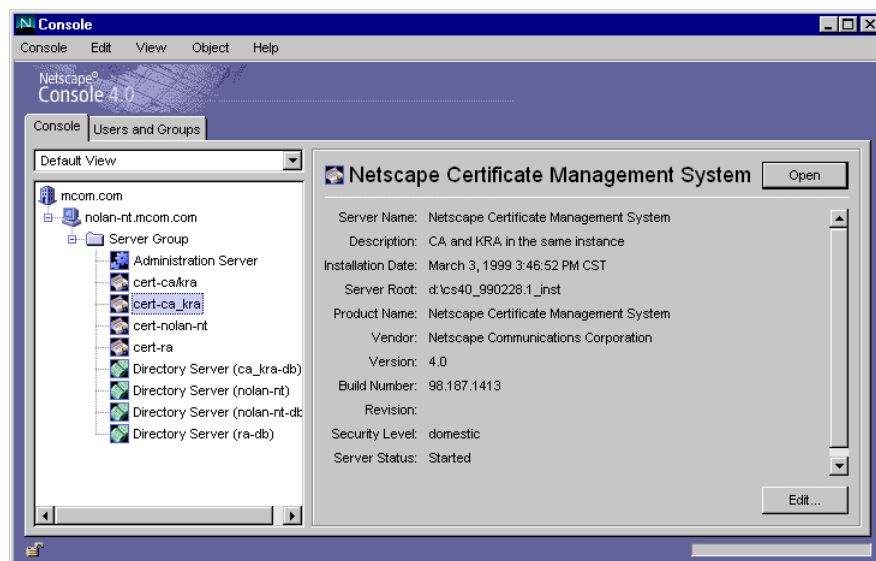
- Netscape Console (page 48)
- Accessing Netscape Console (page 53)
- The CMS Window (page 55)
- Accessing the CMS Window (page 63)

**Note** You can use Netscape Console for managing various network resources. However, this chapter's focus is on using Netscape Console for CMS administration. For complete information about Netscape Console, see *Managing Servers with Netscape Console*, which is included with the CMS documentation.

## Netscape Console

Netscape Console is a stand-alone Java application that provides a GUI-based front end to all network resources registered in an organization's configuration directory. This unified administration interface (shown in Figure 2.1) simplifies network administration by supplying access points to all Netscape version 4.x server instances installed across a network. Similarly, it simplifies basic user and group management by providing a unified administration interface to the user directory.

Figure 2.1 Main Netscape Console window, with a CMS instance selected in the Console tab





## Console Tab

For any given instance of Netscape Console, the limits of the network it can administer are defined by the set of resources whose configuration information is stored in the same configuration directory—that is, the maximum set of hosts and servers that can be monitored from Netscape Console. The *superadministrator* (the person who manages the configuration directory) can set access permissions on all network resources registered in the configuration directory. Thus, for a given administrator using Netscape Console, the actual number of visible servers and hosts may be fewer, depending on the access permissions that administrator has.

The Console tab displays all servers registered in a particular configuration directory, giving you a consolidated view of all the server software and resources under your control. What you control is determined by the access permissions the superadministrator has set up for you.

From this view you can perform tasks across arbitrary groups or a cluster of servers in a single operation. In other words, you can use the Console tab to manage a single server or multiple servers that are installed on different ports on one machine. Also, you can access individual server windows (or administration interfaces) by double-clicking the icons for the corresponding server instance entries (SIEs).

With the exception of Certificate Management System, all server instances displayed on the Console tab store their configuration information in the same configuration directory. For security purposes, Certificate Management System uses file-based configuration which is stored locally on the host system; during installation, the server registers only its SIE in the configuration directory. For details about this file, see “Configuration” on page 67.

You can accomplish various CMS-specific tasks from the Console tab:

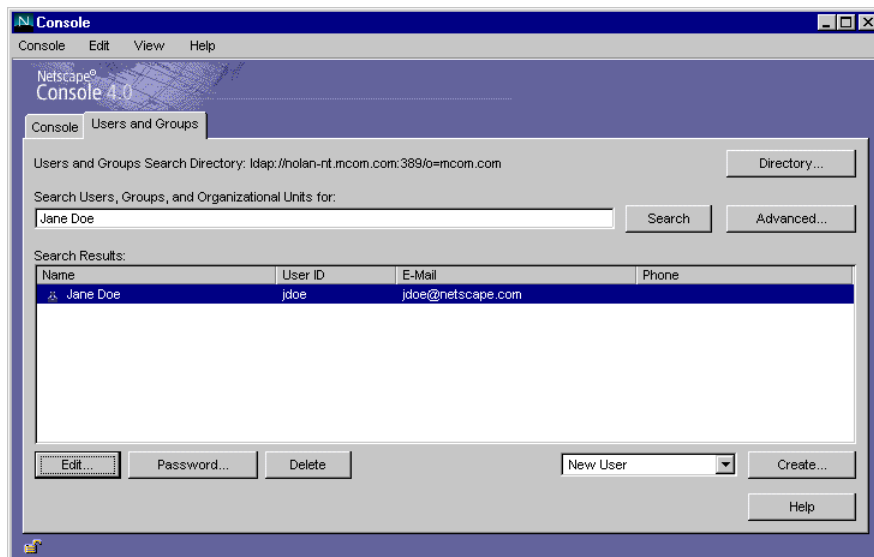
- Install multiple instances of Certificate Management System.
- Remove an instance of Certificate Management System from a system or host.
- Clone an instance of Certificate Management System.
- Set access permissions for Certificate Management System.

- Migrate configuration information from one version of Certificate Management System to another.
- Launch the Administration Server window (so that you can configure an Administration Server instance for administering Certificate Management System).
- Launch the CMS window.

## Users and Groups Tab

The Users and Groups tab (shown in Figure 2.2) manages user accounts, group lists, and access control information for individual users and groups. All applications registered within the Netscape Console framework share core user and group information in the user directory, which typically is a global directory for corporatewide user data.

Figure 2.2 Users and Groups tab of Netscape Console



From this tab, you can accomplish various user- and group-specific tasks, such as these:

- Add, modify, and delete user and group information in the user directory.
- Search for specific user and group entries in the user directory.

## Netscape Administration Server

Netscape Administration Server is a unique, web-based (HTTP) server that enables you to configure all your Netscape version 4.x servers, including Certificate Management System, through Netscape Console. Administration Server must be running before you can configure any of these servers. It is included with all Netscape servers and is installed when you install your first server in a *server group*. A server group refers to servers that are installed in a server root directory and that are managed by a single instance of Netscape Administration Server.

You access Administration Server by entering its URL in the Netscape Console login screen. This URL is based on the computer host name and the port number you chose when you installed Certificate Management System. The format for the URL looks like this:

```
http://<machine_name>.<your_domain>.<domain>:<port_number>
```

Whenever you try to gain access to Administration Server, you will be prompted to authenticate yourself by entering your user ID and password. These are the *administrator* user name and password that you specified when you installed Certificate Management System (or the first server in the server group) and Administration Server on your computer. Once Administration Server is running, you can use Netscape Console to administer all servers in that group, including Certificate Management System.

For complete details about Netscape Administration Server, see *Managing Servers with Netscape Console*.

## Starting Administration Server

The CMS installation program automatically starts the instance of Administration Server that you identified during installation for monitoring Certificate Management System. If you stopped Administration Server after installation, you must start it before you can administer Certificate Management System from the CMS window.

You can start the server from Netscape Console, the command line, or the Windows NT Services panel.

- To start Administration Server from Netscape Console:
  1. Access Netscape Console (see “Accessing Netscape Console” on page 53).
  2. In the Console tab, locate the Administration Server instance that you want to start, and double-click the corresponding entry.

The Administration Server window appears.

3. In the Tasks tab, click Start the Server.

- To start Administration Server from the command line:

At the prompt, enter the following line:

```
<server_root>/admin-<instance_id>/start-admin
```

<server\_root> is the directory where the Administration Server binaries are kept. You first specified this directory during server installation.

<instance\_id> is the ID for this instance of Administration Server. You first specified this during server installation.

This command starts Administration Server at the port number you specified during installation. Once the server is running, you can use Netscape Console to access Certificate Management System.

- Administration Server runs as a service in a Windows NT system. You can use the Windows NT Services panel to start the service directly.

## Shutting Down Administration Server

It is good security practice to shut down Administration Server when you are not using it. This minimizes the chances of someone else changing your configuration.

You can shut down the server from Netscape Console, the command line, or the Windows NT Services panel.

- To shut down Administration Server from Netscape Console:
  1. Access Netscape Console (see “Accessing Netscape Console” on page 53).
  2. In the Console tab, locate the Administration Server instance that you want to shut down, and double-click the corresponding entry.  
The Administration Server window appears.
  3. In the Tasks tab, click Stop the Server.

- To shut down Administration Server from the command line:

At the prompt, enter the following line:

```
<server_root>/admin-<instance_id>/stop-admin
```

`<server_root>` is the directory where the Administration Server binaries are kept. You first specified this directory during server installation.

`<instance_id>` is the ID for this instance of Administration Server. You first specified this during server installation.

- Administration Server runs as a service in a Windows NT system; you can use the Windows NT Services panel to stop the service directly.

## Accessing Netscape Console

You can launch and use Netscape Console only when Netscape Administration Server is running; for information on starting the server, see “Starting Administration Server” on page 52. When you launch Netscape Console, it displays a login window. You are required to authenticate to the configuration

directory by entering your administrator's ID, your password, and the URL (including port number) of the Administration Server representing a server group to which you have access. You cannot use Netscape Console without having login access to at least one server group on your network.

1. Open the Netscape Console application by using the appropriate option:

- Local access in Unix

At the command-line prompt, enter the following line:

```
<server_root>/admin-<instance_id>/start-console
```

<server\_root> is the directory where the Administration Server binaries are kept. You first specified this directory during server installation.

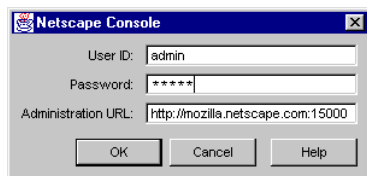
<instance\_id> is the ID for this instance of Administration Server. You first specified this during server installation.

- Local access in Windows NT

Double-click the Netscape Console icon on your desktop; this icon was created when you installed the first Netscape version 4.x server.



The Netscape Console window appears.



2. Authenticate yourself to the configuration directory.

**User ID.** Type the *administrator ID* you specified when you installed Administration Server on your machine. You installed Administration Server either when you installed the first Netscape 4.x server or as a part of CMS installation.

**Password.** Type the *administrator* password that you specified when you installed Administration Server on your computer during CMS installation.

**Administration URL.** This field should show the URL to Administration Server. If it doesn't, type the URL in this field. The URL is based on the computer host name and the Administration Server port number you chose when you installed Certificate Management System. Use this format:

```
http://<machine_name>.<your_domain>.<domain>:<port_number>
```

For example, if your domain name is `mozilla` and you installed Administration Server in a machine called `my_server` and specified port number 12345, the URL would look like this:

```
http://<my_server>.<mozilla>.<com>:12345
```

**3.** Click OK.

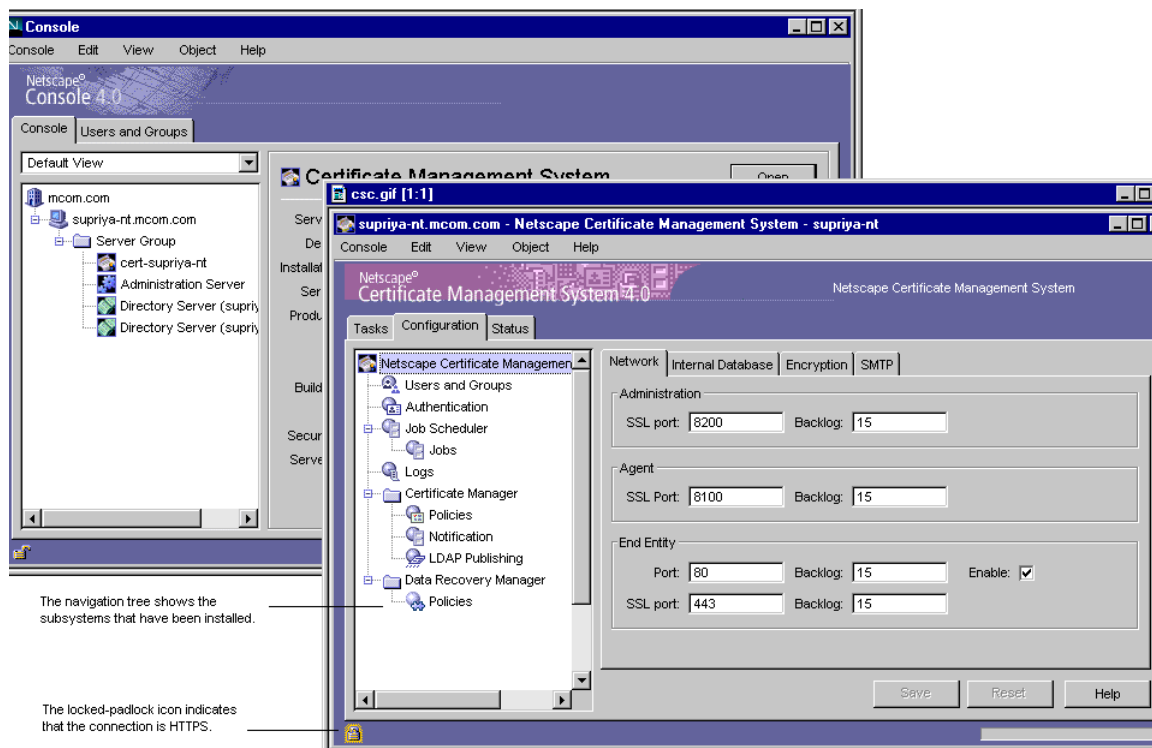
Netscape Console appears with a list of all the servers and resources under your control (see Figure 2.1).

- To view general information about a specific server, click the corresponding entry.
- To access the administration interface for a specific server, double-click the corresponding entry.

## The CMS Window

The CMS window is a GUI-based administration interface that allows you to perform day-to-day operational and managerial duties for Certificate Management System. You launch the CMS window from within Netscape Console (see Figure 2.3). You can use the CMS window to access the server locally or remotely.

Figure 2.3 Certificate Management System window, launched from Netscape Console

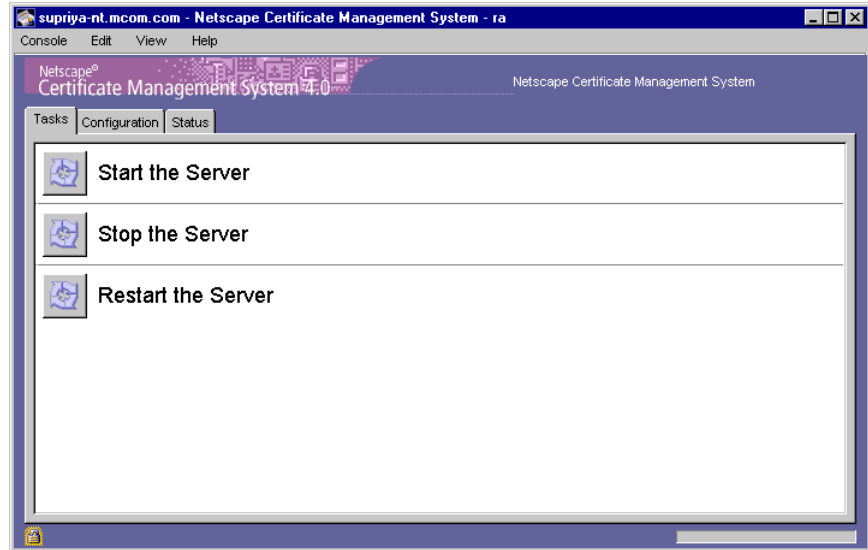


The CMS window has three separate tabs, each addressing specific administrative areas: the Tasks tab, the Configuration tab, and the Status tab.



## Tasks Tab

The Tasks tab allows you to perform frequently required tasks. From this tab, you can start, stop, and restart the server.



For details on these common tasks, see “Stopping Certificate Management System” on page 110 and “Restarting Certificate Management System” on page 113.

# Configuration Tab

The Configuration tab allows you to view the current server configuration settings and change them.

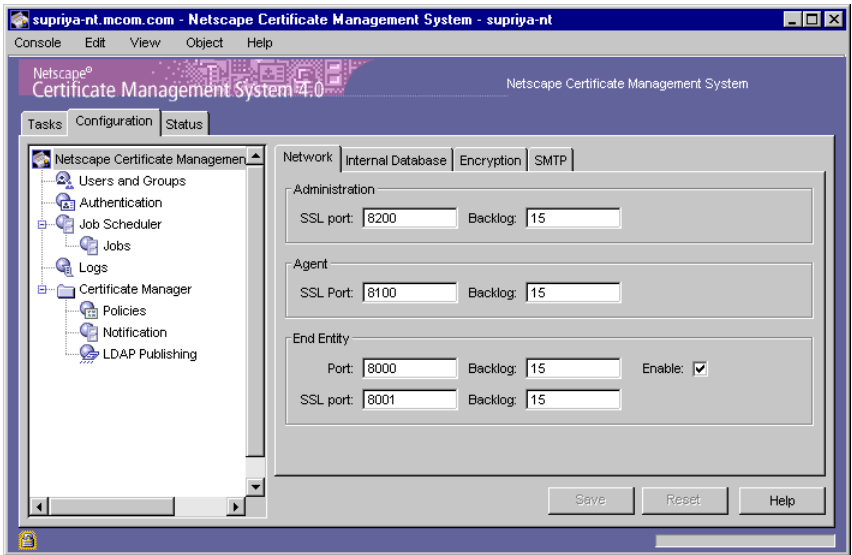


Table 2.1 provides details about the tasks you can accomplish from this tab. You access specific settings by selecting an entry in the navigation tree and working with the tabs that appear in the right pane.

Table 2.1 Tasks you can accomplish from the Configuration tab

Task	Description
Configuring network settings	<p>This involves operations such as the following:</p> <ul style="list-style-type: none"><li>• Changing the administration, agent, and end-entity ports of Certificate Management System</li><li>• Changing the names of CMS instances</li></ul> <p>For details, see “Configuring Ports, Database, and SMTP Settings” on page 121.</p>

Table 2.1 Tasks you can accomplish from the Configuration tab (Continued)

Task	Description
Configuring the internal database settings	This involves specifying the host name and port number of the directory server that Certificate Management System should use for storing data. For details, see “Internal Database” on page 128.
Managing CMS keys and certificates	<p>This involves operations such as the following:</p> <ul style="list-style-type: none"> <li>• Managing the CMS certificate database. For details, see “Managing the Certificate Database” on page 246.</li> <li>• Generating new and renewing existing certificates for the Certificate Manager, Registration Manager, and Data Recovery Manager. For details, see “Getting New Certificates for the Subsystems” on page 232.</li> <li>• Installing new hardware tokens. For details, see “Tokens for Storing Keys and Certificates” on page 192.</li> <li>• Configuring the Data Recovery Manager for archival and recovery of end users’ encryption private keys. For details, see “Recovering Encrypted Data” on page 623.</li> </ul>
Configuring SMTP settings	This involves specifying the host name and port number of the mail server that Certificate Management System should use for sending email notifications. For details, see “SMTP Settings” on page 131.

Table 2.1 Tasks you can accomplish from the Configuration tab (Continued)

Task	Description
Setting up privileged users	<p>This involves operations such as the following:</p> <ul style="list-style-type: none"> <li>• Entering information about privileged users (administrators, agents, and trusted managers) into the CMS internal database. For details, see “Setting Up Privileged Users” on page 149.</li> <li>• Modifying user information. For details, see “Changing Privileged-User Information” on page 175.</li> <li>• Deleting users from the database. For details, see “Deleting a Privileged User” on page 181.</li> </ul>
Determining authentication for end entities	<p>This involves operations such as the following:</p> <ul style="list-style-type: none"> <li>• Viewing currently registered authentication plug-in modules. For details, see “Configuring Authentication for End Entities” on page 309.</li> <li>• Configuring Certificate Management System to use a specific authentication mechanism to authenticate end entities when they request service from the server. For details, see “Configuring Authentication for End Entities” on page 309.</li> <li>• Registering custom authentication plug-in modules. For details, see “Developing Authentication Plug-ins” on page 329.</li> </ul>
Scheduling jobs	<p>This involves operations such as the following:</p> <ul style="list-style-type: none"> <li>• Viewing currently registered plug-in modules for jobs. For details, see “Configuring Jobs” on page 381.</li> <li>• Configuring Certificate Management System to execute specific jobs. For details, see “Configuring Jobs” on page 381.</li> </ul>

Table 2.1 Tasks you can accomplish from the Configuration tab (Continued)

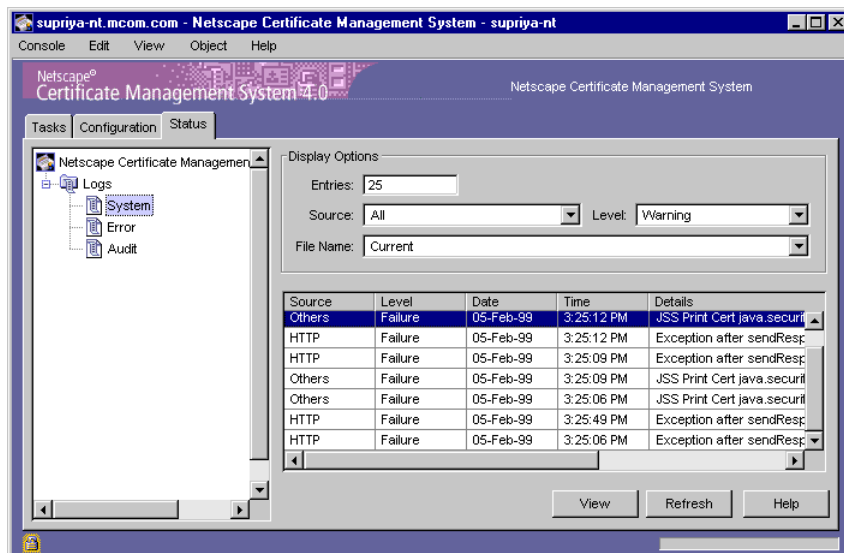
Task	Description
Enabling automated notifications	<p>This involves operations such as the following:</p> <ul style="list-style-type: none"> <li>• Entering the information required by the server to send automated notifications to one or more agents when a request enters the agent queue. For details, see “Notification of New Request in Queue” on page 367.</li> <li>• Entering the information required by the server to send automated certificate-issuance notifications to end entities when the server issues them certificates. For details, see “Notifications of Certificate Issuance to End Entities” on page 365.</li> <li>• Customizing the notification message templates to suit your organization’s requirements. For details, see “Customizing Message Templates” on page 374.</li> </ul>
Configuring certificate issuance and management policies	<p>This involves operations such as the following:</p> <ul style="list-style-type: none"> <li>• Viewing currently registered policy plug-in modules for a Certificate Manager or Registration Manager. For details, see “Configuring Policies” on page 459.</li> <li>• Configuring the Certificate Manager or Registration Manager for certificate formulation, issuance, renewal, and revocation policies, and configuring the Data Recovery Manager for the archiving and recovery of end users’ encryption private keys. For details, see “Configuring Policies” on page 459.</li> </ul>
Publishing certificates and CRLs to an LDAP-compliant directory	<p>This involves operations such as the following:</p> <ul style="list-style-type: none"> <li>• Configuring the Certificate Manager and Registration Manager to publish certificate information to an LDAP-compliant directory, such as Netscape Directory Server. For details, see “Configuring Subsystems for LDAP Publishing” on page 503.</li> <li>• Configuring the Certificate Manager for publishing a certificate revocation list (CRL) to an LDAP-compliant directory, such as Netscape Directory Server. For details, see “Publishing CRLs” on page 523.</li> </ul>

Table 2.1 Tasks you can accomplish from the Configuration tab (Continued)

Task	Description
Enabling end entity and agent interaction	<p>This involves operations such as the following:</p> <ul style="list-style-type: none"> <li>• Enabling or disabling end-entity interaction with the Certificate Manager and Registration Manager. For details, see “Introduction to End-Entity and Agent Interfaces” on page 533.</li> <li>• Customizing the default end entity and agent forms to suit your organization’s requirements. For details, see “Customizing End-Entity and Agent Interfaces” on page 547.</li> </ul>
Managing CMS logs	<p>This involves configuring system, error, and audit logs maintained by Certificate Management System. For details, see “Configuring Logs” on page 581.</p>
Configuring the Data Recovery Manager	<p>This involves configuring the Data Recovery Manager for archival and recovery of end users’ encryption private keys. For details, see “Recovering Encrypted Data” on page 623.</p>
Backing up and restoring CMS data	<p>This involves operations such as the following:</p> <ul style="list-style-type: none"> <li>• Periodically backing up the CMS data.</li> <li>• In the event of data loss, using the resulting archives to restore the data.</li> </ul> <p>For details, see “Backing Up and Restoring Data” on page 665.</p>

## Status Tab

The Status tab allows you to monitor the server by viewing the contents of various logs maintained by Certificate Management System.



You can monitor active as well as rotated log files. For details, see the following sections:

- “Monitoring System Logs” on page 587
- “Monitoring Error Logs” on page 589
- “Monitoring Audit Logs” on page 592

## Accessing the CMS Window

You access the CMS window from Netscape Console. For details on Netscape Console, see “Netscape Console” on page 48.

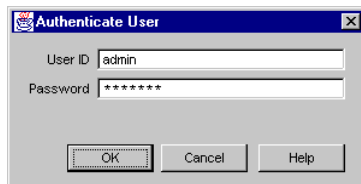
The Console tab of Netscape Console contains a list of network resources that are under your control. In this list you can identify CMS instances by their icons or by server identifiers you specified during installation (for example, you may have named a CMS instance ABC Corp CA).

To open the CMS window for a specific CMS instance:

1. Access Netscape Console (see “Accessing Netscape Console” on page 53).
2. In the Console tab, select the Server Group that contains the CMS instance you want to use as your source.
3. In the navigation tree, locate the CMS instance you want to administer.
4. Select the instance and click Open or double-click the corresponding entry.

If the selected server is not running, you are asked to start the server first. In that case, start the server, and then repeat steps 2 through 4. For information on starting the server, see “Starting Certificate Management System” on page 106.

If the selected server is running, you are prompted to authenticate to Certificate Management System.



5. Enter the appropriate information:

**User ID.** If you are logging in for the first time, type the Certificate Administrator ID; you specified this user ID during installation (so that you could log in to the CMS window without having to create privileged-user entries). Otherwise, type your privileged-user ID (administrator ID).

**Password.** If you are logging in for the first time, type the Certificate Administrator password; you specified this password during installation (so that you could log in to the CMS window without having to create privileged-user entries). Otherwise, type your privileged-user (administrator) password; see



Upon successful authentication, the CMS window appears (Figure 2.3 on page 56).

**Note** Accessing the CMS window is a privileged operation that is restricted to CMS administrators. After you log in for the first time, create at least one user in each of the default groups; see “Groups and Their Privileges” on page 146.



# Configuration

The runtime properties of Netscape Certificate Management System (CMS) are governed by a set of configuration parameters. These parameters are stored in a file that is read by the server during startup.

When you install Certificate Management System, the installer creates an ASCII file, named `CMS.cfg`, and populates it with the appropriate configuration parameters. You can control the way Certificate Management System functions by making the appropriate changes to the configuration information.

This chapter explains how the installation affects the number of configuration files created in your machine and their contents. It also explains ways in which you can modify the configuration and precautions you should take when doing so. The chapter ends with a road map to configuring individual subsystems.

The chapter has the following sections:

- Effects of Installation Type on Configuration (page 68)
- Locating the Configuration File (page 71)
- Modifying the Configuration (page 71)
- Road Map to Configuring Subsystems (page 83)

# Effects of Installation Type on Configuration

For each instance of Certificate Management System there is a configuration file, named `CMF.cfg`. The configuration file controls the runtime properties of the corresponding CMS instance.

A CMS instance can include a single subsystem or two subsystems in one of the following combinations:

- A single Certificate Manager, Registration Manager, or Data Recovery Manager
- A Certificate Manager and Data Recovery Manager together
- A Registration Manager and Data Recovery Manager together

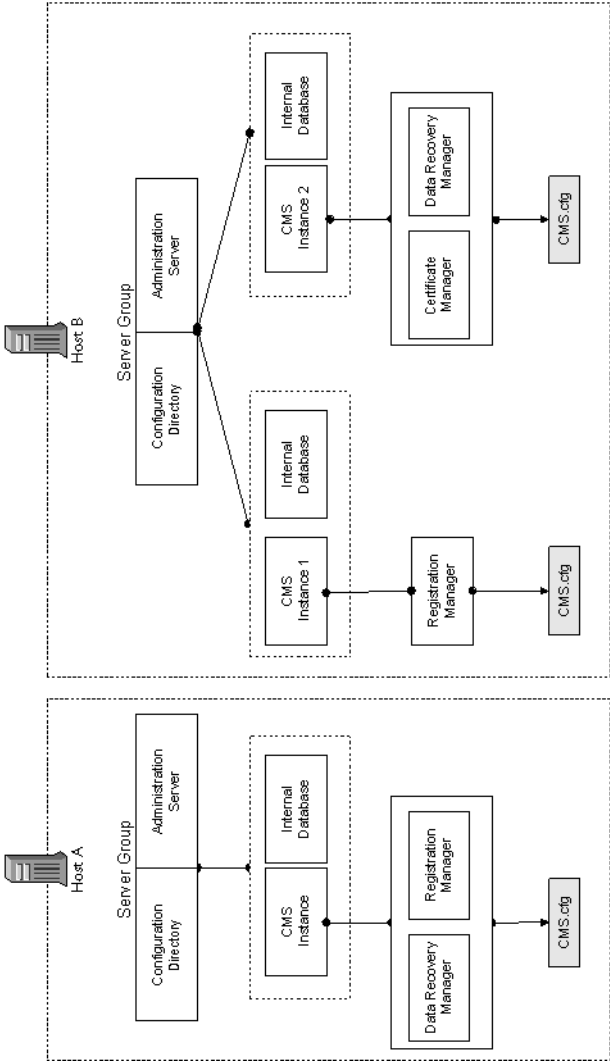
Figure 3.1 on page 69 illustrates a deployment scenario involving two instances of Certificate Management System running on the same host (Host A) and a single instance running on another host (Host B). Notice the two separate configuration files for the instances running on Host A, one for each CMS instance.

Although the names of both the configuration files are the same, the information included in the files differs according to the subsystems installed in each instance. For example, the configuration file for *CMS Instance 1* includes only those parameters that govern the Registration Manager, whereas the configuration file for *CMS Instance 2* includes parameters that control both the Certificate Manager and Data Recovery Manager.

It is also important to understand that subsystems installed in a CMS instance share certain parts of the configuration. They use the same

- Administration, agent, and end-entity ports for interaction
- Internal token and trust database
- SSL ciphers during SSL negotiation
- Privileged users (administrators and agents)
- Log files to log messages
- Internal database for data storage

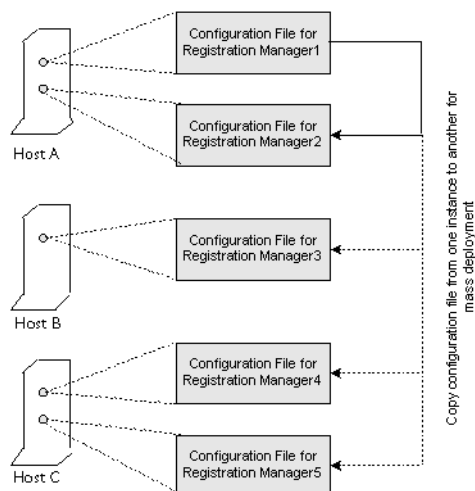
Figure 3.1 How installation affects configuration



## Duplicating a Configuration from One Instance to Another

If you have deployed a large number of CMS instances that are identical—for example, multiple Registration Managers—and you want all these instances to have the same configuration, you can accomplish this by configuring one of the instances and then replacing the configuration files of the other instances with the one that contains the required configuration. Figure 3.2 illustrates this quick way of deploying multiple Registration Managers with the same configuration.

Figure 3.2 Duplicating a configuration



**Caution** Be careful when replacing configuration of one instance with another. The configuration file for an instance contains instance-specific parameters. If you replace these parameters, the instance will fail to start or function properly.

# Locating the Configuration File

Each instance of Certificate Management System has its own configuration file, `CMF.cfg`. The default location for this file is as follows:

```
<server_root>/cert-<instance_id>/config/...
```

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

`<instance_id>` is the ID for this instance of Certificate Management System. You first specified this when you installed this server.

## Modifying the Configuration

You can modify the CMS configuration in two ways:

- By changing the configuration parameter values from the CMS window. This is the recommended method for changing configuration. See “Changing the Configuration from the CMS Window” on page 71.
- By manually changing the configuration parameter values in the configuration file, `CMF.cfg`. See “Changing the Configuration by Editing the Configuration File” on page 72.

## Changing the Configuration from the CMS Window

The CMS window allows you to view the current configuration of a CMS instance and make the required changes. Because this is the recommended method for changing configuration, the chapters that follow focus on explaining how to change the various configuration parameter values from the CMS window.

**Note** You may find the road map provided in “Road Map to Configuring Subsystems” on page 83 useful in setting up your CMS instances.

## Changing the Configuration by Editing the Configuration File

This section explains how to change the CMS configuration by editing the configuration parameter values in the file `CMF.cfg`. This ASCII file is read by Certificate Management System when it is started.

**Caution** Do not edit the configuration file directly if you are not familiar with the configuration parameters or if you are not sure that the changes you intend to make are acceptable by the server. Certificate Management System will fail to start up if you make incorrect modifications to the configuration file. Incorrect configuration can also result in data loss.

Also, before you start editing the configuration file, be sure to read “Guidelines for Editing the Configuration File” on page 73.

To modify the configuration file directly:

1. Stop the CMS instance whose configuration file you want to edit (see “Stopping Certificate Management System” on page 110).
2. Open a terminal window.
3. Locate the configuration file (`CMF.cfg`).

The default location for this file is:

```
<server_root>/cert-<instance_id>/config/...
```

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

`<instance_id>` is the ID for this instance of Certificate Management System. You first specified this when you installed this server.

4. Open the configuration file in an ASCII editor.
5. Edit information in the file and save your changes.
6. Restart Certificate Management System (see “Restarting Certificate Management System” on page 113).



## Guidelines for Editing the Configuration File

The file-based, configuration-store implementation for Certificate Management System is based on `java.util.Properties`. The following guidelines may help you interpret the information in the configuration file.

- The format of the configuration file is as follows:

```
#comment

[parameter]=value

value

[parameter]

multi

line

value (e.g. base-64 encoded certificate)
```

- Comment lines, blank lines, unknown parameters, or misspelled parameters are ignored by Certificate Management System. Comment lines begin with a number sign (#). A line beginning with white space is considered a continuation of the previous line.
- The configuration file has many sections. Some sections contain parameters specific to the subsystems that have been installed; the other sections contain parameters that are shared by the subsystems. Subsystem-specific parameters are distinguished by a prefix identifying the subsystem:
  - `ca` for the Certificate Manager
  - `ra` for the Registration Manager
  - `kra` for the Data Recovery Manager
- The parameter names and their values are strings. The parameter names can be hierarchically structured with `'.'` notation with multiple levels—for example, `ca.Policy.rule.RSAKeyRule.maxSize`. The entries corresponding to a lower level (such as `Policy` in the example) can be requested from the configuration corresponding to its higher level (`ca` in the example).

- The values that need to be localized (such as distinguished names in multibyte format) should be entered in `utf8` format. For more information on this format, see the document *UTF-8, a transformation format of Unicode and ISO 10646*, available at this URL:

`http://info.internet.isi.edu:80/in-notes/rfc/files/rfc2044.txt`

- Certificate Management System writes out the configuration in a sorted order.
- The values of some parameters are referenced to other parts of the configuration file. For example, assume that a parameter is defined as `subsystem.id=ca`; when this parameter is processed by the server, all the parameters beginning with `ca` will be used.

## Sample Configuration File

The following sample configuration is of a Certificate Manager and Data Recovery Manager installed in an instance.

**Important** This sample file includes some of the parameters used by Certificate Management System. However, there is no guarantee that an arbitrary set of options you create will work.

```
_000=##
_001=## File Created On      : Thu Apr 29 12:43:05 PDT 1999
_002=##

instanceRoot=C:/Netscape/Server4/cert-testCA-KRA

agentGateway._000=##
agentGateway._001=## Agent Gateway
agentGateway._002=##
    agentGateway.docRoot=C:/Netscape/Server4/cert-testCA-KRA/web/agent
    agentGateway.enableAdminEnroll=true
    agentGateway.enableBulkInterface=true
    agentGateway.enableConnector=true
    agentGateway.mimeTypeConf=C:/Netscape/Server4/
        cert-testCA-KRA/config/mime.types
    agentGateway.numServices=1
    agentGateway.service0=https
    agentGateway.CAGetBySerial.successTemplate=/ca/ImportCert.template
```

```

agentGateway.adminEnroll.successTemplate=/ca/EnrollSuccess.template
agentGateway.bulkissuance.errorTemplate=/ca/bulkissuance.template
agentGateway.bulkissuance.pendingTemplate=/ca/bulkissuance.template
agentGateway.bulkissuance.rejectedTemplate=/ca/bulkissuance.template
agentGateway.bulkissuance.successTemplate=/ca/bulkissuance.
    template
agentGateway.bulkissuance.svcpendingTemplate=/ca/
    bulkissuance.template
agentGateway.bulkissuance.unauthorizedTemplate=/ca/
    bulkissuance.template
agentGateway.bulkissuance.unexpectedErrorTemplate=/ca/
    bulkissuance.template
agentGateway.https.backlog=15
agentGateway.https.nickName=Server-Cert cert-testCA-KRA
agentGateway.https.port=8100
agentGateway.https.type=https

auths._000=##
auths._001=## new authentication
auths._002=##
auths.impl._000=##
auths.impl._001=## authentication manager implementations
auths.impl._002=##
    auths.impl.UidPwdDirAuth.class=com.netscape.certsrv.
        authentication.UidPwdDirAuthentication
    auths.impl.UidPwdPinDirAuth.class=com.netscape.certsrv.
        authentication.UidPwdPinDirAuthentication
auths.revocationChecking.bufferSize=5
auths.revocationChecking.ca=ca
auths.revocationChecking.enabled=true
auths.revocationChecking.kra=kra
auths.revocationChecking.validityInterval=300

ca.id=ca
ca.local=true

ca.Policy.order=KeyAlgRule, RSAKeyRule, DefaultValidityRule,
    DefaultRenewalValidityRule, DefaultRevocationRule, NSCertTypeExt,
    KeyUsageExt, SubjectKeyIdentifierExt, AuthorityKeyIdentifierExt,
    BasicConstraintsExt
ca.Policy.impl._000=##
ca.Policy.impl._001=## Policy Implementations
ca.Policy.impl._002=##

```

```

ca.Policy.impl.AuthorityKeyIdentifierExt.class=com.netscape.certsrv.
policy.AuthorityKeyIdExt
ca.Policy.impl.BasicConstraintsExt.class=com.netscape.certsrv.
policy.BasicConstraintsExt
ca.Policy.impl.DSAKeyConstraints.class=com.netscape.certsrv.policy.
DSAKeyConstraints
ca.Policy.impl.DefaultRevocation.class=com.netscape.certsrv.policy.
DefaultRevocation
ca.Policy.impl.KeyAlgorithmConstraints.class=com.netscape.certsrv.
policy.KeyAlgorithmConstraints
ca.Policy.impl.KeyUsageExt.class=com.netscape.certsrv.policy.
KeyUsageExt
ca.Policy.impl.NSCertTypeExt.class=com.netscape.certsrv.policy.
NSCertTypeExt
ca.Policy.impl.RSAKeyConstraints.class=com.netscape.certsrv.
policy.RSAKeyConstraints
ca.Policy.impl.RenewalValidityConstraints.class=com.netscape.
certsrv.policy.RenewalValidityConstraints
ca.Policy.impl.SubjectKeyIdentifierExt.class=com.netscape.
certsrv.policy.SubjectKeyIdExt
ca.Policy.impl.ValidityConstraints.class=com.netscape.certsrv.
policy.ValidityConstraints
ca.Policy.rule.AuthorityKeyIdentifierExt.enable=true
ca.Policy.rule.AuthorityKeyIdentifierExt.implName=
AuthorityKeyIdentifierExt
ca.Policy.rule.AuthorityKeyIdentifierExt.predicate=
ca.Policy.rule.BasicConstraintsExt.enable=true
ca.Policy.rule.BasicConstraintsExt.implName=BasicConstraintsExt
ca.Policy.rule.BasicConstraintsExt.predicate=
ca.Policy.rule.DSAKeyRule.enable=true
ca.Policy.rule.DSAKeyRule.implName=DSAKeyConstraints
ca.Policy.rule.DSAKeyRule.maxSize=2048
ca.Policy.rule.DSAKeyRule.minSize=512
ca.Policy.rule.DSAKeyRule.predicate=
ca.Policy.rule.DefaultRenewalValidityRule.enable=true
ca.Policy.rule.DefaultRenewalValidityRule.implName=
RenewalValidityConstraints
ca.Policy.rule.DefaultRenewalValidityRule.maxValidity=365
ca.Policy.rule.DefaultRenewalValidityRule.minValidity=30
ca.Policy.rule.DefaultRenewalValidityRule.predicate=
ca.Policy.rule.DefaultRenewalValidityRule.renewalInterval=15
ca.Policy.rule.DefaultRevocationRule.enable=true
ca.Policy.rule.DefaultRevocationRule.implName=DefaultRevocation
ca.Policy.rule.DefaultRevocationRule.predicate=

```

```

ca.Policy.rule.DefaultValidityRule.enable=true
ca.Policy.rule.DefaultValidityRule.implName=ValidityConstraints
ca.Policy.rule.DefaultValidityRule.maxValidity=365
ca.Policy.rule.DefaultValidityRule.minValidity=30
ca.Policy.rule.DefaultValidityRule.predicate=
ca.Policy.rule.KeyAlgRule.algorithm=RSA
ca.Policy.rule.KeyAlgRule.enable=true
ca.Policy.rule.KeyAlgRule.implName=KeyAlgorithmConstraints
ca.Policy.rule.KeyAlgRule.predicate=
ca.Policy.rule.KeyUsageExt.enable=true
ca.Policy.rule.KeyUsageExt.implName=KeyUsageExt
ca.Policy.rule.KeyUsageExt.predicate=
ca.Policy.rule.NSCertTypeExt.enable=true
ca.Policy.rule.NSCertTypeExt.implName=NSCertTypeExt
ca.Policy.rule.NSCertTypeExt.predicate=
ca.Policy.rule.RSAKeyRule.enable=true
ca.Policy.rule.RSAKeyRule.exponents=3,7,17,65537
ca.Policy.rule.RSAKeyRule.implName=RSAKeyConstraints
ca.Policy.rule.RSAKeyRule.maxSize=2048
ca.Policy.rule.RSAKeyRule.minSize=512
ca.Policy.rule.RSAKeyRule.predicate=
ca.Policy.rule.SubjectKeyIdentifierExt.enable=true
ca.Policy.rule.SubjectKeyIdentifierExt.implName=
    SubjectKeyIdentifierExt
ca.Policy.rule.SubjectKeyIdentifierExt.predicate=certType==ca

ca.connector.KRA.id=kra
ca.connector.KRA.local=true

ca.crl._000=##
ca.crl._001=## CA CRL
ca.crl._002=##
    ca.crl.MasterCRL.allowExtensions=false
    ca.crl.MasterCRL.autoUpdateInterval=0
    ca.crl.MasterCRL.class=com.netscape.certsrv.ca.CRLIssuingPoint
    ca.crl.MasterCRL.description=com.netscape.certsrv.ldap.
        LdapCrlPublisher
    ca.ldappublish.certPublish.class=com.netscape.certsrv.ldap.
        LdapUserCertPublisher
    ca.ldappublish.crlPublish.class=com.netscape.certsrv.ldap.
        LdapCrlPublisher
    ca.ldappublish.type.ca.mapper.baseDN=
    ca.ldappublish.type.ca.mapper.class=com.netscape.certsrv.ldap.

```

```

    LdapCertCompsMap
ca.ldappublish.type.ca.mapper.dnComps=0
ca.ldappublish.type.ca.mapper.filterComps=CN
ca.ldappublish.type.ca.publisher.class=com.netscape.certsrv.ldap.
    LdapCaCertPublisher
ca.ldappublish.type.client.mapper.baseDN=
ca.ldappublish.type.client.mapper.class=com.netscape.certsrv.ldap.
    LdapCertCompsMap
ca.ldappublish.type.client.mapper.dnComps=0
ca.ldappublish.type.client.mapper.filterComps=UID,CN
ca.ldappublish.type.client.publisher.class=com.netscape.certsrv.
    ldap.LdapUserCertPublisher
    ca.ldappublish.type.crl.mapper.baseDN=
ca.ldappublish.type.crl.mapper.class=com.netscape.certsrv.ldap.
    LdapCrlIssuerCompsMap
ca.ldappublish.type.crl.mapper.dnComps=0
ca.ldappublish.type.crl.mapper.filterComps=CN
ca.ldappublish.type.crl.publisher.class=com.netscape.certsrv.ldap.
    LdapCrlPublish

ca.notification.certIssued.emailSubject=Your Certificate Request
ca.notification.certIssued.emailTemplate=C:/Netscape/Server4/
    cert-testCA-KRA/emails/certIssued_CA.html
ca.notification.certIssued.enabled=true
ca.notification.certIssued.senderEmail=cert_central@netscape.com
ca.notification.requestInQ.emailSubject=Certificate Request in Queue
ca.notification.requestInQ.emailTemplate=C:/Netscape/Server4/
    cert-testCA-KRA/emails/reqInQueue.html
ca.notification.requestInQ.enabled=true
ca.notification.requestInQ.recipientEmail=CA_agent@netscape.com
ca.notification.requestInQ.senderEmail=CA_admin@netscape.com
ca.signing.cacertnickname=caSigningCert cert-testCA-KRA
ca.signing.defaultSigningAlgorithm=MD5withRSA
ca.signing.tokenname=Internal Key Storage Token

dbs.ldap=internaldb

eeGateway._000=##
eeGateway._001=## End Entity Gateway
eeGateway._002=##
    eeGateway.authority=ca
    eeGateway.docRoot=C:/Netscape/Server4/cert-testCA-KRA/web/ee
    eeGateway.dynamicVariables=serverdate=serverdate(), subsystemname=

```

```

        subsystemname()
eeGateway.enableConnector=true
eeGateway.mimeTypeConf=C:/Netscape/Server4/cert-testCA-KRA/
    config/mime.types
eeGateway.numServices=2
eeGateway.service0=http
eeGateway.service1=https
eeGateway.http.backlog=15
eeGateway.http.enable=true
eeGateway.http.port=80
eeGateway.http.type=http
eeGateway.https.backlog=15
eeGateway.https.nickName=Server-Cert cert-testCA-KRA
eeGateway.https.port=443
eeGateway.https.type=https

internaldb._000=##
internaldb._001=## Internal Database
internaldb._002=##
    internaldb.basedn=o=NetscapeCertificateServer
    internaldb.maxConns=10
    internaldb.minConns=3
    internaldb.ldapauth.authType=BasicAuth
    internaldb.ldapauth.bindDN=cn=Directory Manager
    internaldb.ldapauth.bindPWPrompt=Internal LDAP Database
    internaldb.ldapconn.host=myHost.netscape.com
    internaldb.ldapconn.port=38900
    internaldb.ldapconn.secureConn=false

jobsScheduler._000=##
jobsScheduler._001=## jobsScheduler
jobsScheduler._002=##
    jobsScheduler.enabled=true
    jobsScheduler.interval=1
    jobsScheduler.impl.RenewalNotificationJob.class=com.netscape.
        certsrv.jobs.RenewalNotificationJob
    jobsScheduler.impl.RequestInQueueJob.class=com.netscape.certsrv.
        jobs.RequestInQueueJob
    jobsScheduler.impl.UnpublishExpiredJob.class=com.netscape.certsrv.
        jobs.UnpublishExpiredJob
    jobsScheduler.job.certRenewalNotifier.cron=0 3 * * 1-5
    jobsScheduler.job.certRenewalNotifier.emailSubject=Certificate
        Renewal Notification

```

```

jobsScheduler.job.certRenewalNotifier.emailTemplate=C:/Netscape/
  Server4/cert-testCA-KRA/emails/rnJob1.txt
jobsScheduler.job.certRenewalNotifier.enabled=false
jobsScheduler.job.certRenewalNotifier.notifyEndOffset=30
jobsScheduler.job.certRenewalNotifier.notifyTriggerOffset=30
jobsScheduler.job.certRenewalNotifier.pluginName=
  RenewalNotificationJob
jobsScheduler.job.certRenewalNotifier.senderEmail=
jobsScheduler.job.certRenewalNotifier.summary.emailSubject=
  Certificate Renewal Notification Summary
jobsScheduler.job.certRenewalNotifier.summary.emailTemplate=
  C:/Netscape/Server4/cert-testCA-KRA/emails/rnJob1Summary.txt
jobsScheduler.job.certRenewalNotifier.summary.enabled=true
jobsScheduler.job.certRenewalNotifier.summary.itemTemplate=
  C:/Netscape/Server4/cert-testCA-KRA/emails/rnJob1Item.txt
jobsScheduler.job.certRenewalNotifier.summary.recipientEmail=
jobsScheduler.job.certRenewalNotifier.summary.senderEmail=
jobsScheduler.job.requestInQueueNotifier.cron=0 0 * * 0
jobsScheduler.job.requestInQueueNotifier.enabled=false
jobsScheduler.job.requestInQueueNotifier.pluginName=
  RequestInQueueJob
jobsScheduler.job.requestInQueueNotifier.subsystemId=ca
jobsScheduler.job.requestInQueueNotifier.summary.emailSubject=
  Requests in Queue Summary Report
jobsScheduler.job.requestInQueueNotifier.summary.emailTemplate=
  C:/Netscape/Server4/cert-testCA-KRA/emails/riq1Summary.html
jobsScheduler.job.requestInQueueNotifier.summary.enabled=true
jobsScheduler.job.requestInQueueNotifier.summary.recipientEmail=
jobsScheduler.job.requestInQueueNotifier.summary.senderEmail=
jobsScheduler.job.unpublishExpiredCerts.cron=0 0 * * 6
jobsScheduler.job.unpublishExpiredCerts.enabled=false
jobsScheduler.job.unpublishExpiredCerts.pluginName=
  UnpublishExpiredJob
jobsScheduler.job.unpublishExpiredCerts.summary.emailSubject=Expired
  Certificates Unpublished Summary
jobsScheduler.job.unpublishExpiredCerts.summary.emailTemplate=C:/
  Netscape/Server4/cert-testCA-KRA/emails/euJob1.html
jobsScheduler.job.unpublishExpiredCerts.summary.enabled=true
jobsScheduler.job.unpublishExpiredCerts.summary.itemTemplate=C:/
  Netscape/Server4/cert-testCA-KRA/emails/euJob1Item.html
jobsScheduler.job.unpublishExpiredCerts.summary.recipientEmail=
jobsScheduler.job.unpublishExpiredCerts.summary.senderEmail=

jss._000=##

```



```

jss._001=## JSS
jss._002=##
    jss.certdb=C:/Netscape/Server4/cert-testCA-KRA/config/cert7.db
    jss.enable=true
    jss.keydb=C:/Netscape/Server4/cert-testCA-KRA/config/key3.db
    jss.moddb=C:/Netscape/Server4/admin-serv/config/secmod.db
    jss.ssl.cipherfortezza=true
    jss.ssl.cipherpref=
    jss.ssl.cipherversion=cipherdomestic

kra.storageUnit.certdb=C:/Netscape/Server4/cert-testCA-KRA/
    config/kra-cert.db
kra.storageUnit.keydb=C:/Netscape/Server4/cert-testCA-KRA/
    config/kra-key.db
kra.storageUnit.mn=C:/Netscape/Server4/cert-testCA-KRA/
    config/kra-mn.conf
kra.storageUnit.nickName=kraStorageCert
kra.transportUnit.nickName=kraTransportCert cert-testCA-KRA

logAudit._000=##
logAudit._001=## Logging
logAudit._002=##
    logAudit.bufferSize=512
    logAudit.expirationTime=2592000
    logAudit.fileName=C:/Netscape/Server4/cert-testCA-KRA/logs/audit
    logAudit.flushInterval=300
    logAudit.level=3
    logAudit.maxFileSize=100
    logAudit.on=true
    logAudit.rolloverInterval=2592000
logError._000=##
logError._001=## Logging
logError._002=##
    logError.bufferSize=512
    logError.expirationTime=2592000
    logError.fileName=C:/Netscape/Server4/cert-testCA-KRA/logs/error
    logError.flushInterval=300
    logError.level=3
    logError.maxFileSize=100
    logError.on=true
    logError.rolloverInterval=2592000
logNTAudit.NTEventSourceName=cert-testCA-KRA

```

```

logNTAudit.level=2
logNTAudit.on=true
logNTSystem.NTEventSourceName=cert-testCA-KRA
logNTSystem.level=2
logNTSystem.on=true
logSystem._000=##
logSystem._001=## Logging
logSystem._002=##
    logSystem.bufferSize=512
    logSystem.expirationTime=2592000
    logSystem.fileName=C:/Netscape/Server4/cert-testCA-KRA/logs/system
    logSystem.flushInterval=300
    logSystem.level=3
    logSystem.maxFileSize=100
    logSystem.on=true
    logSystem.rolloverInterval=2592000

oidmap.challenge_password.class=com.netscape.certsrv.cmsgateway.
    cert.crs.ChallengePassword
oidmap.challenge_password.oid=1.2.840.113549.1.9.7
oidmap.extensions_requested.class=com.netscape.certsrv.cmsgateway.
    cert.crs.ExtensionsRequested
oidmap.extensions_requested.oid=2.16.840.1.113733.1.9.8

os.serverName=cert-testCA-KRA
os.userid=nobody

radm._000=##
radm._001=## Remote Admin
radm._002=##
    radm.keepAliveOn=true
    radm.mimeTypeConf=C:/Netscape/Server4/cert-testCA-KRA/
        config/mime.types
    radm.numServices=1
    radm.service0=https
    radm.https.backlog=15
    radm.https.maxThreads=10
    radm.https.minThreads=3
    radm.https.nickName=Server-Cert cert-testCA-KRA
    radm.https.port=8200
    radm.https.timeout=10
    radm.https.type=https

```

```

smtp.host=localhost
smtp.port=25

subsystem._000=##
subsystem._001=## Loadable Subsystems
subsystem._002=##
    subsystem.0.class=com.netscape.certsrv.kra.KeyRecoveryAuthority
    subsystem.0.id=kra
    subsystem.1.class=com.netscape.certsrv.ca.CertificateAuthority
    subsystem.1.id=ca
    subsystem.2.class=com.netscape.certsrv.cmsgateway.EEGateway
    subsystem.2.id=eeGateway

usrgrp._000=##
usrgrp._001=## User/Group
usrgrp._002=##
    usrgrp.ldap=internaldb

```

## Road Map to Configuring Subsystems

This section outlines how to configure an instance of Certificate Management System and indicates where to find the information required to accomplish the task.

### Step 1. Check Which Subsystems are Installed in the Instance

Open the CMS window for the CMS instance you installed, and check the navigation tree to see which subsystems are installed in that instance; this way you will know the subsystems you should configure. To open the CMS window, see “Accessing the CMS Window” on page 63.

### Step 2. Check the Port Numbers

Check the port numbers assigned for administration, agent, and end-entity operations. Make the appropriate modifications, if necessary. Keep in mind that all subsystems installed in an instance use the same ports. To change the ports, see “CMS Ports” on page 121.

### Step 3. Verify Key Pair and Certificates

When you install a CMS instance, the server prompts you to create the certificates required for the subsystems in that instance to function. You should check the certificates used by each subsystem, see if you need to get additional certificates, use hardware tokens, etc.

- Each subsystem in an instance has a set of certificates that it uses for specific purposes. Understand how and when the subsystem uses its certificates. For details, see “Keys and Certificates for the Main Subsystems” on page 184.
- Determine if you want to generate any new certificates. For example, if you have two subsystems installed in an instance, you may want them to use separate SSL server certificates; by default, there’s only one SSL server certificate per instance. For details, see “Getting New Certificates for the Subsystems” on page 232.
- Determine if you want to use hardware tokens for generating and storing these certificates. If required, install new hardware tokens. For details, see “Tokens for Storing Keys and Certificates” on page 192.
- Determine if you want to renew any of the existing certificates. For example, if you have issued certificates with very short validity periods, you might want to renew them. For details, see “Renewing Certificates for the Subsystems” on page 239.
- Check the certificate database to see which CA certificates are trusted. Delete any unwanted CA certificates, change the trust settings of CA certificates that you don’t want to trust to *untrusted*, and install any new CA certificate or certificate chains. For details, see “Managing the Certificate Database” on page 246.

### Step 4. Check the SMTP Settings

Check the mail server settings—Certificate Management System uses this information to send automated email notifications. If necessary, make the appropriate changes to the host name and port number. Keep in mind that all subsystems installed in an instance use the same mail server. To change the mail server-specific information, see “SMTP Settings” on page 131.

## Step 5. Set up Privileged Users

Set up required administrators and agents. This way you can delegate administration and agent tasks to other individuals. For details, see “Setting Up Privileged Users” on page 149.

If you have installed remote Registration Managers that have certificates signed by third-party CAs (not a Certificate Manager) you should add their certificates to the Certificate Manager’s database to facilitate SSL client authenticated communication. For details, see “Setting Up Trusted Managers” on page 158.

## Step 6. Customize End Entity and Agent Forms

End entities can interact with the Certificate Manager and Registration Manager with the help of end-entity forms; end entities cannot interact with the Data Recovery Manager. Similarly, agents can interact with the appropriate subsystem using the agent forms. Certificate Management System provides HTML forms-based interfaces for end entities and agents out of the box. For details, see “Introduction to End-Entity and Agent Interfaces” on page 533.

Determine which forms you want to use for end-entity enrollment and whether they require any customization. You may also use your own forms for this purpose, provided you add the required JavaScript. For details, see “End-Entity Services” on page 533.

When customizing end-entity forms, keep in mind the authentication scheme—manual- or directory-based—you want to employ for your end entities.

## Step 7. Setup Authentication for End Entities

Depending on your PKI setup, you may need to do this for a Certificate Manager or Registration Manager, or for both. For example, you may have a set up in which Registration Managers act as front ends to Certificate Managers—that is, end entities interact with Registration Managers only; they do not interact with the Certificate Manager.

- I. Determine whether you want to use any of the directory-based authentication plug-in modules provided out of the box. For details, see “End-Entity Authentication During Certificate Enrollment” on page 265. If you don’t, you either have to use the manual mode or will have to plug-in

your own modules. For information on developing authentication plug-in modules and registering them in the CMS framework, see “End-Entity Authentication During Certificate Enrollment” on page 265.

2. Configure the Certificate Manager or Registration Manager to use a specific authentication module. For details, see “Adding an Authentication Instance” on page 317.
3. Update the end-entity enrollment forms to use this authentication mechanism.
4. Enable end-entity interaction with the subsystem:
  - To enable end-entity interaction for a Certificate Manager, see “Enabling End-Entity Interaction with a Certificate Manager” on page 538.
  - To enable end-entity interaction for a Registration Manager, see “Enabling End-Entity Interaction with a Registration Manager” on page 539.

## Step 8. Schedule Jobs

Each CMS instance includes a *Job Scheduler* component that can execute specific jobs at specified times. The Job Scheduler functions similar to a traditional Unix *cron* daemon in that it takes registered cron jobs and executes them at a preconfigured date and time. For details, see “Introduction to Job Scheduling and Notifications” on page 345.

During installation, a few jobs are already created and enabled. Jobs that you might want to schedule include email notifications of timed events (such as the expiration of a certificate) that require action on the part of users, and periodic activities such as removing expired certificates from the publishing directory.

1. Check each job and decide whether you want to use it. If you don't, you can either disable it or delete it altogether from the configuration. For those jobs that you want to use, check the configuration parameter values and make changes as appropriate. For details, see “Modifying a Job” on page 391.
2. Determine if you want to add any new jobs. Check the built-in job plug-in modules to see if they can be used to create the jobs you want. For details, see “Built-in Job Plug-in Modules” on page 346. You can also plug in your own job modules in the CMS framework and use them.
3. Add new jobs, if any. For details, see “Adding a Job” on page 387.

## **Step 9: Enable Event-Driven Notifications**

You can also configure both Certificate Manager and Registration Manager to send email notifications automatically to end entities, agents, or administrators when certain events occur. Unlike jobs that are executed at preconfigured schedule, these notifications are event-driven—that is, whenever an event occurs, the server notifies the user. Notifiable events include certificate issuance and pending requests in an agent queue.

Decide if you want to turn on any of the notifications. If you do, the server uses the mail server specified in the SMTP settings (Step 4) to send these notifications. For details, see “Event-Driven Notifications” on page 364.

## **Step 10. Set up Policies**

Each subsystem in a CMS instance has its own policy processor. If you have installed more than one subsystem in an instance, you should apply the instructions in this section to each subsystem. That is, you should configure the Certificate Manager or Registration Manager for certificate formulation, issuance,

renewal, and revocation policies. Similarly, configure the Data Recovery Manager for key archival and recovery policies. To understand policy, see “Introduction to Policy” on page 401.

1. During installation, a few policy rules are already created and enabled. Check each policy rule and decide whether you want to use it. If you don't, you can either disable it or delete it altogether from the configuration. For those rules that you want to use, check the configuration parameter values and make changes as appropriate. For details, see “Modifying a Policy Rule” on page 473.
2. Determine if you want to add any new policy rules. Check the built-in policy plug-in modules to see if they can be used to create the rules you want. For details, see “Built-in Policy Plug-in Modules” on page 409. You can also plug-in your own modules in the CMS framework and use them.
3. Add new rules, if any. For details, see “Adding a Policy Rule” on page 468.

## **Step II. Set up LDAP Publishing**

This step is optional, and is applicable to the Certificate Manager and Registration Manager only—you need to do this only if you want the Certificate Manager or Registration Manager to publish certificates and CRLs to an LDAP-compliant directory, such as Netscape Directory Server.

1. Determine if you want to use any of the mapper and publisher classes provided out of the box. See “Directory Update Process” on page 490. Otherwise, plug in your own modules in the CMS framework and use them.
2. Set up the directory for publishing. See “Setting Up the Directory for Publishing” on page 504.
3. Configure the Certificate Manager or Registration Manager to publish certificate information to an LDAP-compliant directory using specific mapper and publisher plug-in modules.
  - To configure the Certificate Manager, see “Configuring a Certificate Manager for LDAP Publishing” on page 507.
  - To configure the Registration Manager, see “Configuring a Registration Manager for LDAP Publishing” on page 515.



4. Configure the Certificate Manager to publish CRL information to an LDAP-compliant directory; the Registration Manager cannot publish CRLs. See “Updating CRLs Automatically” on page 526.

## **Step 12. Set up Logging**

Each instance of Certificate Management System maintains extensive audit, error, and system logs. By looking at these logs, you can monitor a server's activities. Also, by configuring these logs, you can control the information that gets written to the log files. Because Certificate Management System maintains the log files in the file system of the host machine, it is important that you configure the logs appropriately (so that the host machine doesn't get overloaded). Be sure to read “Introduction to Logs” on page 567; this chapter will help you decide log configuration.

Once you decide the configuration for server logs, follow the information in “Configuring Logs” on page 581 and configure all the three logs. Then, start monitoring the server's activities as explained in “Monitoring Logs” on page 586.

## **Step 13. Set up archival and recovery for end users' keys**

If you have installed the Data Recovery Manager, follow the instructions in “Setting Up Key Archival and Recovery Process” on page 641 and set up archival and recovery for end users' encryption private keys.

## **Step 14. Test your PKI Setup**

Use the information in “Issuing and Managing End-Entity Certificates” on page 603 and test that the certificate issuance, renewal, and revocation operations work satisfactorily.

If you have deployed the Data Recovery Manager, follow the information in “Step 5. Test Your Key Archival Setup” on page 650 and “Step 6. Test Your Key Recovery Setup” to test the key archival and recovery operation respectively.

## **Step 15. Plan for Backing up CMS Configuration and Data**

It is a good practice to periodically back up the CMS data on to some backup media. Creating backups will help you use them for data restoration in the event of data loss. For details, see “Backing Up and Restoring Data” on page 665.



# 2

## *Managing Certificate Management System*

*Chapter 4*    Installing and Uninstalling Instances

*Chapter 5*    Starting and Stopping Instances



# Installing and Uninstalling Instances

After the initial installation of Netscape Certificate Management System (CMS), you may need to install additional instances, remove unwanted instances, or duplicate configuration in multiple instances. This chapter describes how to manage these tasks by using Netscape Console, the single, unified administration interface for your network.

You can use Netscape Console only when Netscape Administration Server is running. During CMS installation, you specified a port number for the Administration Server instance you will use to administer Certificate Management System. If Administration Server is shut down, be sure to start it at this port. To minimize security risks, shut down the Administration Server when you have finished using Netscape Console. For more information about Netscape Console, see “Administration Tasks and Tool” on page 47.

The chapter has the following sections:

- Installing Multiple Instances (page 94)
- Viewing Instance Information (page 96)
- Changing the Name of an Instance (page 98)
- Removing an Instance from a System (page 99)
- Uninstalling Certificate Management System (page 101)

# Installing Multiple Instances

Multiple instances of Certificate Management System can run on the same machine. You might, for example, install multiple Registration Managers, all reporting to the same Certificate Manager, to handle requests from different types of users (end users, servers, and routers) or from users from different domains. For example deployment scenarios, see the *Netscape Certificate Management System Installation and Deployment Guide*.

Once Certificate Management System is installed on a machine, you can use that CMS installation to create multiple instances of the server on the same machine. Administration Server contains all the files necessary to install another instance of Certificate Management System on the same machine; you don't have to run the complete installation (setup) program again. However, you do need to run the CMS installation wizard each time you create an instance, so that you can configure the server and generate the required certificates. So, before attempting to create another instance of Certificate Management System, be sure to read about the installation wizard in the *Netscape Certificate Management System Installation and Deployment Guide*.

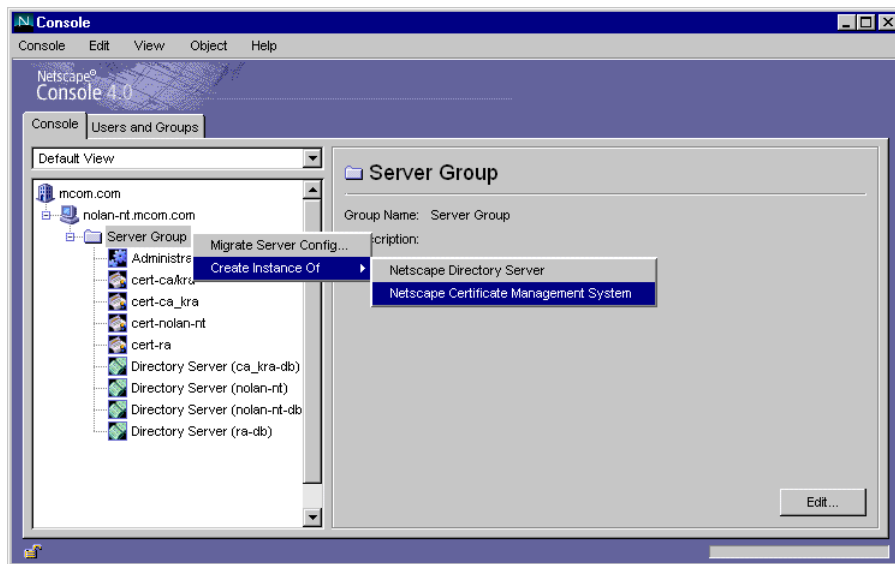
When you install additional CMS instances on the same machine, you are required to specify different ports for each CMS instance to listen on. For example, you will have to set up one server to listen on port 443 and another server to listen on port 4430. However, if you install multiple CMS instances on a machine that has been set up with more than one IP addresses, you can configure each instance to listen to a specific IP address—this enables you to use same the port numbers for different CMS instances installed on the same machine.

Keep in mind that when you create a new instance, you'll be required to specify different port numbers; the installation wizard allows you to specify the port numbers only, not IP addresses. After you have successfully created the instance, you can edit the CMS configuration file to specify the IP addresses for individual CMS ports and then change the port numbers. For details on editing the configuration file to include the IP addresses, see “Specifying IP Addresses for CMS Instances” on page 127. For details on changing the port numbers, see “Configuring Port Numbers” on page 125.

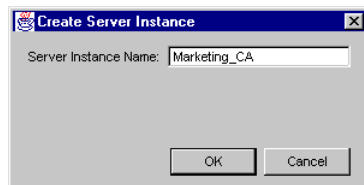
To create another instance of Certificate Management System with a separate configuration file (and certificates):

1. Access Netscape Console (see “Accessing Netscape Console” on page 53).
2. In the Console tab, select the server group that contains the CMS instance you want to use as your source.
3. From the Object menu, choose the Create Instance Of option and, in the pop-up menu that appears, choose Netscape Certificate Management System.

As shown in this figure, you can also right-click to choose this option from the pop-up menu.



The Create Server Instance dialog box appears.



4. Enter a name for the instance. For the name, you can use any combination of letters (aA to zZ), digits (0 to 9), an underscore (\_), and a hyphen (-); other characters and spaces are not allowed. For example, you can type `Netscape_root-CA` as the instance name, but not `Netscape root CA`.
5. Click OK.

The instance you created appears in the navigation tree. Note that the instance name appears in this form:

`cert-<instance_name>`

`<instance_name>` is the name you specified for the new CMS instance.

For example, if you named the instance `Marketing_CA`, the instance name in the navigation tree will be `cert-Marketing_CA`.

6. To start the installation wizard, double-click the new instance in the navigation tree.

Use the installation wizard to finish configuring the new instance.

## Viewing Instance Information

In Netscape Console, you can view some of the basic information—the name and version number of the server, the directory in which it's installed, and date it was installed—about a CMS instance.

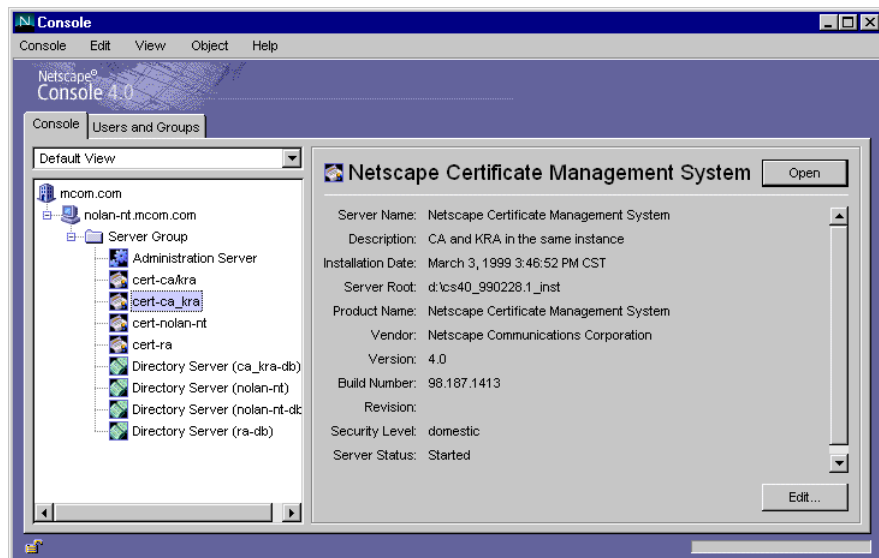
To view information pertaining to a specific CMS instance:

1. Access Netscape Console (see “Accessing Netscape Console” on page 53).
2. In the Console tab, double-click the server group that contains the CMS instance you want to view.



3. In the list of server instances, select the CMS instance you want to view.

The right pane shows information about the selected CMS instance.



The information displayed includes the following:

**Server Name.** A descriptive name of the CMS instance. You can change this name; see “Changing the Name of an Instance” on page 98).

**Description.** Additional information that helps you identify the CMS instance. You can change this description; see “Changing the Name of an Instance” on page 98.

**Installation Date.** The date the server was installed.

**Server Root.** The directory that holds all the files for the selected CMS instance, the files of its Administration Server, and the files of any other Netscape servers in the same server group (that is, administered by that Administration Server). A host typically has only one server root, but more than one is possible, especially if different version numbers of the same server are installed on a single host.

The default server root in Unix is `usr/netscape/server4/` and in Windows NT is `C:\Netscape\Server4`.

**Product Name.** The complete product name.

**Vendor.** The name of the vendor.

**Version.** The version number.

**Build Number.** The number that identifies the build that was used for this installation.

**Security Level.** The server's security level—whether the server is meant for use in the United States (domestic) or any other part of the world (export). (See “Configuring the Server's Security Preferences” on page 223.)

**Server Status.** The server's status—whether it is started or stopped.

## Changing the Name of an Instance

Following installation, the name of a CMS instance is in the form:

`cert-<instance_id>`

`<instance_id>` is the ID for this instance of Certificate Management System. You first specified this when you installed this server.

For example, if you installed an instance of Certificate Management System with an ID of `testCA`, the instance name will be `cert-testCA`.

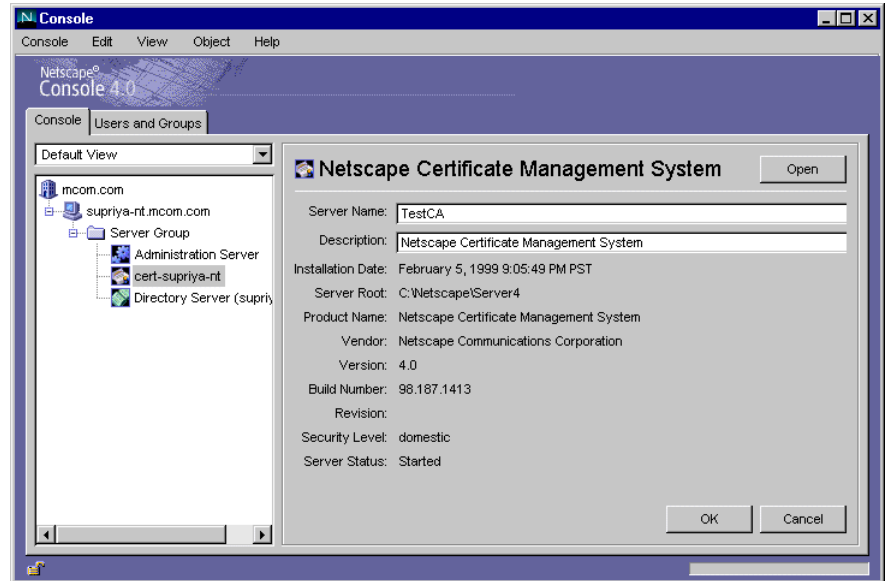
You can change the instance name to a more descriptive one later. If you change the instance name, Certificate Management System uses the new name as a descriptive nickname for the instance. It shows the new name in Netscape Console only; it does not change the original instance ID in the configuration.

To change the name of a particular CMS instance:

1. Access Netscape Console (see “Accessing Netscape Console” on page 53).
2. In the Console tab, select the CMS instance you want to rename.

3. Click Edit.

Details about the selected CMS instance appear in the right pane.



4. Specify the appropriate information:

**Server Name.** Type a descriptive name for the server.

**Description.** Type any additional description for the server. For example, you may want to type information that will help you identify this instance of Certificate Management System.

5. Click OK.

You are returned to the previous screen. The new name appears in the right pane.

## Removing an Instance from a System

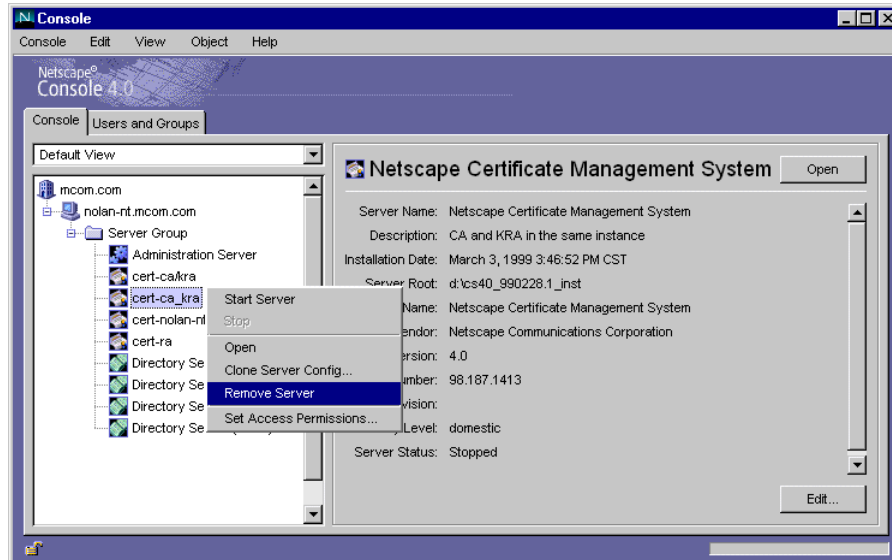
If you are sure you won't need a particular CMS instance anymore, you can use Netscape Console to remove the server instance from your machine. Removing a CMS instance is not the same as uninstalling Certificate Management System;

when you uninstall Certificate Management System, its program files are deleted from the host machine. (For instructions, see “Uninstalling Certificate Management System” on page 101.)

To remove a CMS instance from your machine:

1. Access Netscape Console (see “Accessing Netscape Console” on page 53).
2. In the Console tab, select the CMS instance you want to remove.
3. From the Object menu, choose Stop; you can also right-click to choose this option from the pop-up menu (see the figure below).
4. When the server has stopped, from the Object menu, choose Remove Server.

As shown in the figure below, you can also right-click to choose this option from the pop-up menu.



5. When prompted, confirm that you want to remove the server instance.

The selected CMS instance and the corresponding internal database is removed. The Directory Server (configuration directory) and Administration Server binaries are not removed; you require these to administer the remaining servers installed in the same server group.

# Uninstalling Certificate Management System

To remove files pertaining to Certificate Management System from a host system, run the uninstallation program. Uninstalling Certificate Management System removes all the corresponding CMS instances from the navigation tree of Netscape Console. To remove a specific CMS instance, follow the instructions provided in “Removing an Instance from a System” on page 99.

You can uninstall Certificate Management System in two ways:

- From the command line (locally only)
- On a Windows NT system, by using the Windows NT Add/Remove Programs Utility

## Uninstalling from the Command Line

To uninstall Certificate Management System from the command line:

1. Open a terminal window to your server.
2. In a Unix system, log in either as `root` or using the server's user account (if that is how you started the server).
3. At the command-line prompt, enter the following line:

**Windows NT**      `<server_root>\uninst`

**Unix**              `<server_root>/uninstall`

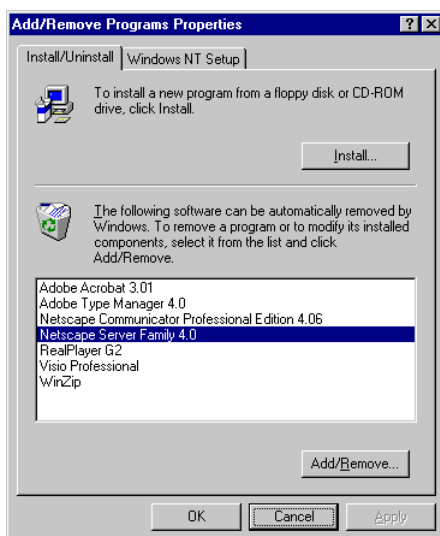
`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

The uninstallation program starts.

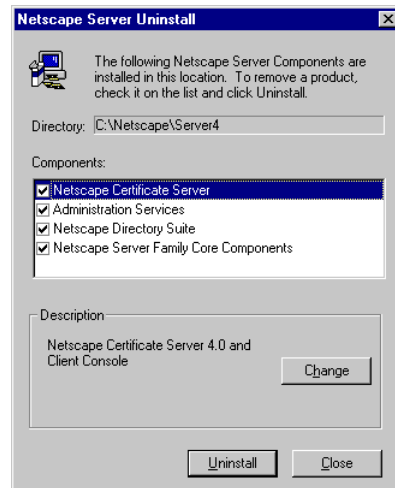
## Uninstalling by Using the Windows NT Add/Remove Programs Utility

To remove Certificate Management System by using the Windows NT Add/Remove Programs utility:

1. From the Start menu, choose Settings, then Control Panel.
2. In the Control Panel, choose Add/Remove Programs.
3. In the Add/Remove Programs Properties window, choose Netscape Server 4.0 Family, and click Add/Remove.



4. In the Netscape Server Uninstall window, make sure all the components are selected, and click Uninstall.



The uninstallation program starts.





# Starting and Stopping Instances

This chapter describes how to start, stop, and restart Netscape Certificate Management System (CMS) and how to check its current status. The chapter also explains the CMS watchdog process, a native bootstrapping program that enables Certificate Management System to start up with a single password instead of multiple ones.

The chapter has the following sections:

- Starting Certificate Management System (page 106)
- Stopping Certificate Management System (page 110)
- Restarting Certificate Management System (page 113)
- Checking System Status (page 115)
- Attending to an Unresponsive Server (page 116)
- CMS Watchdog Process (page 117)

**Note** You can use the CMS window only when the appropriate Administration Server is running. Be sure to start Administration Server at the port you specified during CMS installation. To minimize security risks, shut down Administration Server when you have finished using Netscape Console. For instructions on starting and shutting down Administration Server, see “Netscape Administration Server” on page 51.

# Starting Certificate Management System

Once Certificate Management System is installed, it runs constantly, listening for and accepting requests. You can start Certificate Management System in several ways:

- From Netscape Console (locally and remotely)
- From the command line (locally only)
- On a Windows NT system, from the Windows NT Services panel

## Required Start-up Information

When you start Certificate Management System, you are prompted to enter the *single sign-on* password you specified during installation. This password enables the CMS watchdog (see “CMS Watchdog Process” on page 117) to retrieve all the passwords required by the server to start. These include the following:

- Passwords for the internal or external (if any are currently installed) tokens; these tokens contain certificates and corresponding public and private key pairs for the server.
- The bind password used by Certificate Management System to access and update the internal database.
- The bind password used by Certificate Management System to access and remove PINs from the authentication directory, if you’ve configured Certificate Management System to remove PINs from the authentication directory (see the description for the `ldap.ldapauthbindDN` and `ldap.ldapauth.bindPWPrompt` parameters on Table 9.2 on page 277).
- The bind password used by Certificate Management System to access and update the LDAP directory; this is required only if you have configured Certificate Management System for publishing certificates and CRLs to an LDAP-compliant directory.

You first specified these passwords when you installed Certificate Management System. Keep in mind that the passwords you provide for the tokens unlock a combination of the following private keys:

- If you have installed a Certificate Manager in the currently selected CMS instance, the token password unlocks the private keys for the Certificate Manager's *CA signing* and *SSL server* certificates.
- If you have installed a Registration Manager in the currently selected CMS instance, the token password unlocks the private keys for the Registration Manager's *signing* and *SSL server* certificates.
- If you have installed a Data Recovery Manager in the currently selected CMS instance, the token password unlocks the private keys for the Data Recovery Manager's *storage* keys and *transport* and *SSL server* certificates.

For more information about the CMS keys and certificates, see “Keys and Certificates” on page 183.

**Note** During CMS installation the watchdog stores all the passwords, required by the server for starting up, in a password cache. The cache is maintained in a file encrypted using the single sign-on password you specify during installation. When you change any of the required passwords or provide new passwords, you must start the server from the command-line (see “Starting from the Command Line” on page 109) so that the watchdog can prompt you for the new passwords in order to update the cache.

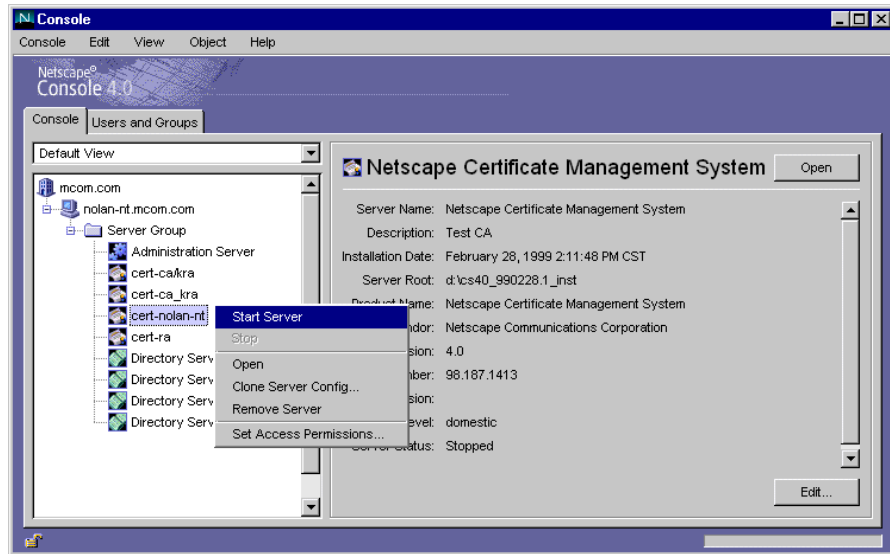
## Starting from Netscape Console

You can use Netscape Console to start an instance of Certificate Management System running on a local or remote host.

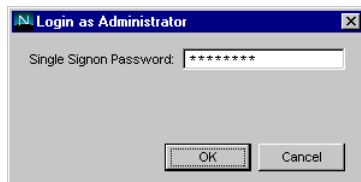
To start Certificate Management System from Netscape Console:

1. Access Netscape Console (see “Accessing Netscape Console” on page 53).
2. In the Console tab, select the Server Group that contains the CMS instance you want to start.
3. In the navigation tree, locate the CMS instance you want to start.

4. Select the instance, right-click, and choose the Start Server option from the pop-up menu.



When you start Certificate Management System, you are prompted to supply the single sign-on password for the server.



5. Enter the single sign-on password you specified during installation and click OK.

Certificate Management System won't start until you provide this password. For more information, see "Required Start-up Information" on page 106.

## Starting from the Command Line

To start Certificate Management System from the command line:

1. Open a terminal window to your server.
2. In a Unix system, log in as `root` if the server runs on ports less than 1024; otherwise, log in either as `root` or with the server's user account.
3. At the command-line prompt, enter the following line:

```
<server_root>/cert-<instance_id>/start-cert[.bat]
```

`.bat` specifies the file extension; this is required only when running the utility on a Windows NT system.

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

`<instance_id>` is the ID for this instance of Certificate Management System. You first specified this when you installed this server.

4. When prompted, supply the single sign-on password.

Certificate Management System won't start until you provide this password. For more information, see "Required Start-up Information" on page 106.

**Note** If Certificate Management System is already running, the start-up command fails. Stop the server first using the `stop-cert` command, then use the `start-cert` command.

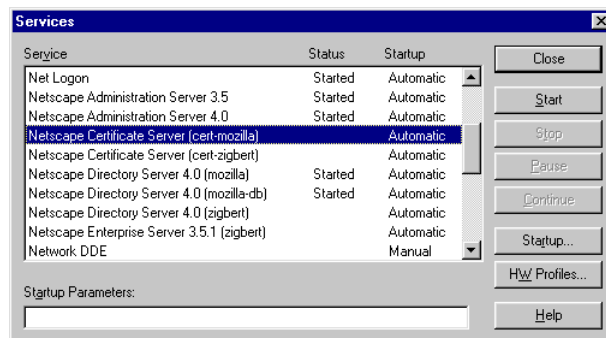
## Starting from the Windows NT Services Panel

If you have installed Certificate Management System on a Windows NT system, you can start the server (as a service) from the Windows NT Services panel (see Figure 5.1). The CMS service has the following name:

```
Netscape Certificate Management System (cert-<instance_id>)
```

`<instance_id>` is the ID for this instance of Certificate Management System. You first specified this when you installed this server.

Figure 5.1 CMS service in the Windows NT Services panel



To start Certificate Management System from the Windows NT Services panel:

1. Click the Start button on your desktop.
2. Select Control Panel from Settings.
3. In the Control Panel window that appears, click Services.
4. Select the CMS instance and click Start.

You are prompted to supply the single sign-on password for the server.

5. Enter the single sign-on password you specified during installation and click OK.

Certificate Management System won't start until you provide this password. For more information, see "Required Start-up Information" on page 106.

## Stopping Certificate Management System

You can stop Certificate Management System in several ways:

- From Netscape Console (locally and remotely)
- From the command line (locally only)
- On a Windows NT system, from the Windows NT Services panel

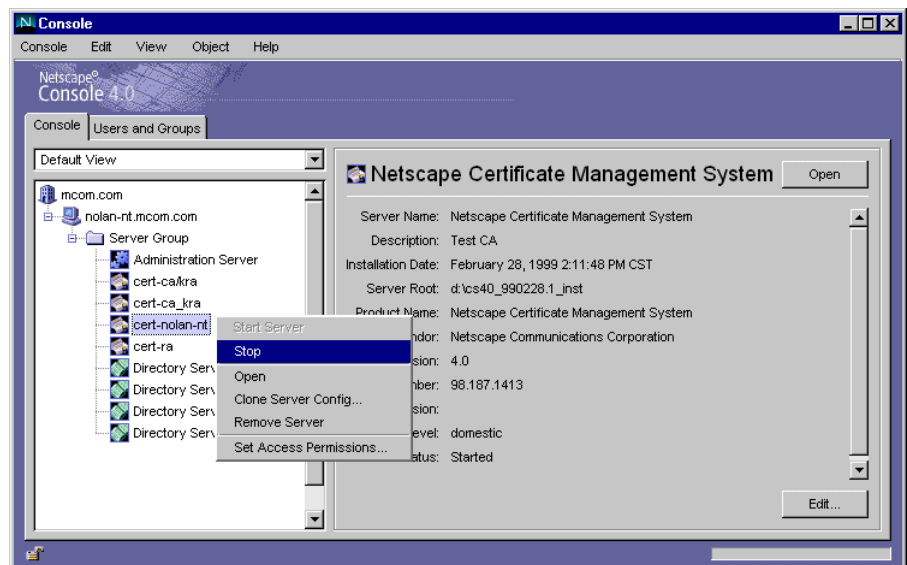
Stopping Certificate Management System shuts down all the subsystems completely, interrupting service until the server is started again. If your machine crashes or is taken offline, the server stops, and any requests it was servicing are lost. You need to start the server again to restore service.

## Stopping from Netscape Console

You can use Netscape Console to stop an instance of Certificate Management System running on a local or remote host.

To stop Certificate Management System from Netscape Console:

1. Access Netscape Console (see “Accessing Netscape Console” on page 53).
2. In the Console tab, select the Server Group that contains the CMS instance you want to stop.
3. In the navigation tree, locate the CMS instance you want to stop.
4. Select the instance, right-click, and choose the Stop option from the pop-up menu.



The server is stopped.

## Stopping from the Command Line

You can stop a CMS instance running on a local host by entering the appropriate command at the command prompt.

To stop a Certificate Management System from the command line:

1. Open a terminal window to your server.
2. In a Unix system, log in either as `root` or using the server's user account (if that is how you started the server).
3. At the command-line prompt, enter the following line:

```
<server_root>/cert-<instance_id>/stop-cert[.bat]
```

`.bat` specifies the file extension; this is required only when running the utility on a Windows NT system.

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

`<instance_id>` is the ID for this instance of Certificate Management System. You first specified this when you installed this server.

The server is stopped.

## Stopping from the Windows NT Services Panel

You can stop a CMS instance running on a local host by stopping the corresponding service; it is identified by the following in the Windows NT Services panel (see Figure 5.1 on page 110):

Netscape Certificate Management System (`cert-<instance_id>`)

`<instance_id>` is the ID for this instance of Certificate Management System. You first specified this when you installed this server.



To stop Certificate Management System from the Windows NT Services panel:

1. Click the Start button on your desktop.
2. Select Control Panel from Settings.
3. In the Control Panel window that appears, click Services.
4. Select the CMS instance and click Stop.
5. When prompted, click Yes.

The server is stopped.

## Restarting Certificate Management System

Whenever you change the CMS configuration, you must save your changes (by clicking the Save button) for the changes to take effect. Some configuration changes also require that you *restart* the server after you save the changes. If restarting is required, the server prompts you accordingly.

You can restart the server in two ways:

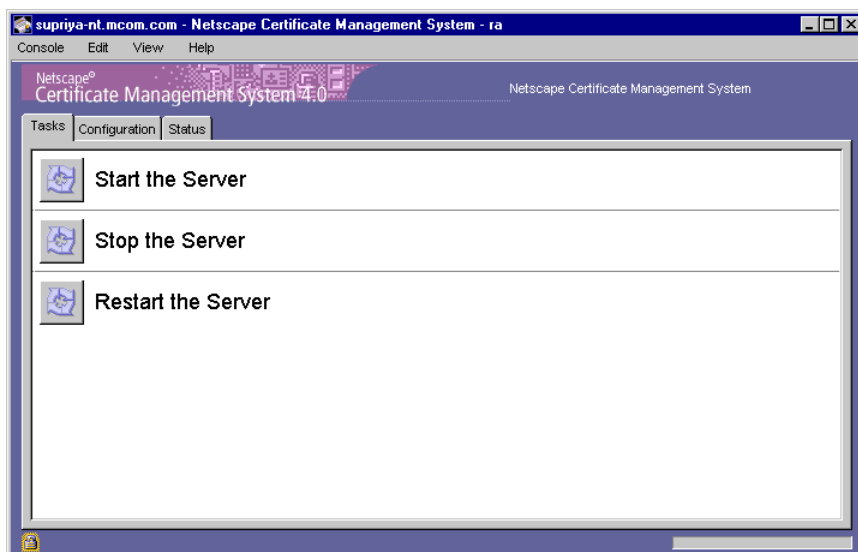
- From the CMS window (locally and remotely)
- From the command line (locally only)

### Restarting from the CMS Window

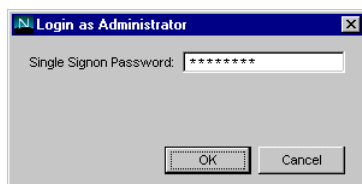
You can use the CMS window to restart an instance of Certificate Management System on a local or remote host.

To restart Certificate Management System from the CMS window:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. In the Tasks tab, click Restart the Server.



When you restart Certificate Management System, you are prompted to supply the single sign-on password for the server.



3. Enter the single sign-on password you specified during installation and click OK.

Certificate Management System won't restart until you provide this password. For more information, see “Required Start-up Information” on page 106.

## Restarting from the Command Line

To restart Certificate Management System from the command line:

1. Open a terminal window to your server.
2. In a Unix system, log in either as `root` or using the server's user account (if that is how you started the server).
3. At the command-line prompt, enter the following line:

```
<server_root>/cert-<instance_id>/restart-cert[.bat]
```

`.bat` specifies the file extension; this is required only when running the utility on a Windows NT system.

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

`<instance_id>` is the ID for this instance of Certificate Management System. You first specified this when you installed this server.

4. When prompted, supply the single sign-on password.

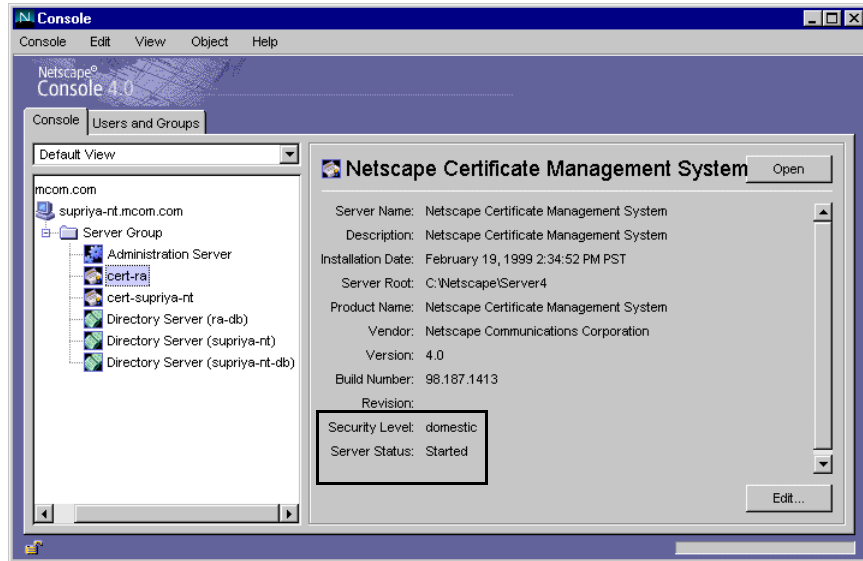
Certificate Management System won't restart until you provide this password. For more information, see "Required Start-up Information" on page 106.

## Checking System Status

You can use Netscape Console to find out whether a particular instance of Certificate Management System is running.

1. Access Netscape Console (see "Accessing Netscape Console" on page 53).
2. In the Console tab, select the SIE that corresponds to the CMS instance you want to check.

In the right pane, check the Server Status field. If the selected instance of Certificate Management System is running, the status will be *Started*. Otherwise it will be *Stopped*.



## Attending to an Unresponsive Server

If an error causes Certificate Management System to become unresponsive, and all attempts to stop it from the UI fail, it may be necessary to kill the server processes manually. The processes that should be killed are the Java virtual machines, called `jssjava`. These processes will be listed in the Windows NT Task Manager. However, because they are system processes, you cannot terminate them from the Task Manager. Instead, you should use the `killproc` command-line tool. This tool is located with the rest of the command-line tools provided with Certificate Management System:

```
<server_root>/bin/cert/tools/...
```

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

In order to kill system processes, the user that runs `killproc` must have the *Debug Programs* permission. By default, this permission is given only to the Administrators group, although this can be changed in the Windows NT User Manager. Assuming it is not changed, `killproc` must be run by a member of the Administrators group (such as the user Administrator).

The `killproc` command takes one argument, the process ID of the process to be killed:

```
killproc <process_id>
```

You can obtain the process ID from the Windows NT Task Manager. For example, to kill the `jssjava` process whose process ID is 255, you should type:

```
c:\> killproc 255

Killed process 255.

c:\>
```

**Note** The `killproc` tool should only be used as a last resort. Because it forces the process to terminate abruptly, the process is not able to cleanup or to save its internal state before exiting.

## CMS Watchdog Process

The CMS watchdog is a native bootstrapping program that provides specific native functions. It works with Certificate Management System to enable it to start up using a single password—instead of multiple passwords—called the *single sign-on* password. In addition, it manages the start-up, stop, and restart states of Certificate Management System.

The watchdog process implements the following operations:

- Starts Certificate Management System (and the Virtual Java Machine, or Java VM).

The watchdog allows you to start Certificate Management System by using a single password instead of the multiple passwords that would otherwise be required. For details on these passwords, see “Required Start-up Information” on page 106.

During CMS installation the watchdog stores all the passwords required by the server for starting up in a password cache. The cache is maintained in a file encrypted using the single sign-on password you specify during installation. When you change any of the required passwords or provide new passwords, you must start the server from the command-line (see “Starting from the Command Line” on page 109) so that the watchdog can prompt you for the new passwords in order to update the cache.

- Stops Certificate Management System.
- Restarts Certificate Management System (after configuration changes).
- Detects Certificate Management System crashes and restarts the server.

The watchdog monitors Certificate Management System and the Java VM, restarting the server in the case of a failure.

- In the Unix version of Certificate Management System, the watchdog records the server process ID (`pid`) and sets the user ID (`uid`) of the process.

# 3

## *System-Level Configuration*

*Chapter 6*    Configuring Ports, Database, and SMTP Settings

*Chapter 7*    Managing Privileged Users and Groups

*Chapter 8*    Keys and Certificates





# Configuring Ports, Database, and SMTP Settings

Subsystems installed in an instance of Netscape Certificate Management System (CMS) share certain configuration information. They use the same administration, agent, and end-entity ports; internal database for data storage; mail server for automated notifications; internal token and trust database for PKI operations; SSL ciphers during SSL negotiation; privileged users; and log files to log messages to. This chapter explains how to configure these ports, the internal database, and the mail server settings for a CMS instance.

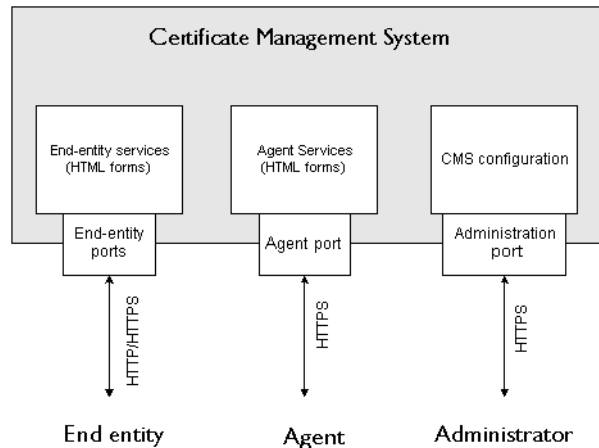
The chapter has the following sections:

- CMS Ports (page 121)
- Internal Database (page 128)
- SMTP Settings (page 131)

## CMS Ports

Certificate Management System listens to different ports for requests from different users. As illustrated in Figure 6.1, it listens to the administration port, the agent port, and end-entity ports.

Figure 6.1 CMS ports for administration, agent, and end-entity operations



When choosing ports for Certificate Management System, be sure to choose ports that are unique on the host system—that is, no other application can be using, or attempting to use, the port numbers you assign to Certificate Management System. To verify that a port is available for use, check the appropriate file for your operating system. Port numbers for network-accessible services are maintained in the file named `services`.

- On a Windows NT system, see `SYSTEM32\DRIVERS\ETC\SERVICES`.
- On a Unix system, see `/etc/services`. (If you are not running as `root` or `superuser` when you install or start the server, you will have to use a port number higher than 1024.)

## Remote Administration Port

The administration port is an SSL (encrypted) port at which Certificate Management System listens to requests from its administration interface; administrators make these requests from the CMS window. When you install Certificate Management System, it assigns a random number (greater than 1024) as the administration port number. You can change this port number at any time, to any number between 1 and 65535. For security reasons you should consider changing the administration port number periodically.

## Agent Port

The agent port is an SSL (encrypted) port at which Certificate Management System listens to requests from agents; agents make these requests from the appropriate Agent Services interface.

- The Certificate Manager and Registration Manager agents use the agent port to process certificate issuance and management requests from end entities and to perform certain other privileged operations over HTTPS.
- Data Recovery Manager agents use the agent port for recovering end users' encryption private keys over HTTPS.

Agent functions always require SSL client authentication. For a list of supported agent operations, see “Agent Services” on page 541.

When you install Certificate Management System, it assigns a random number (greater than 1024) as the agent port number and prompts you to change it, if necessary; the port number can be any number between 1 and 65535. The number you choose for the agent port affects your agent users—all agents access Certificate Management System by specifying the name of the server (the CMS instance) and the agent port number in the URL. For example, if you choose port number 4430, the URL would look like this:

```
https://<host_name>:4430/<subsystem>
```

<host\_name> is in the form <machine\_name>.<your\_domain>.<domain>

<subsystem> is a prefix identifying the subsystem that hosts the agent interface:

- ca for the Certificate Manager
- ra for the Registration Manager
- kra for the Data Recovery Manager

For example, the URL to a Certificate Manager agent interface would look like this:

```
https://testCA.netscape.com:5600/ca
```

If you change the agent port number, be sure to inform your agent users.

## End-Entity Ports

For requests from end entities, Certificate Management System can listen to two ports, an SSL (encrypted) port and a non-SSL port. End entities make these requests from the end entity services interface; see “End-Entity Services” on page 533.

Certificate Management System provides the following services through the HTTP and HTTPS ports:

- The HTTP port can be used to service end-entity-initiated PKI requests, such as enrollment, renewal, and revocation; enrollment requests can include requests from Cisco routers (using the CEP protocol). You have the choice of keeping this port enabled or disabled.
- The HTTPS port can be used to provide the following services for enforcing data privacy and client authentication:
  - End-entity-initiated PKI requests, such as enrollment, renewal, and revocation
  - General certificate retrieval requests, such as retrieving a single certificate identified by a serial number, listing certificates based on certain criteria (for example, an LDAP search filter defined over standard attributes), and getting a CA's certificate chain

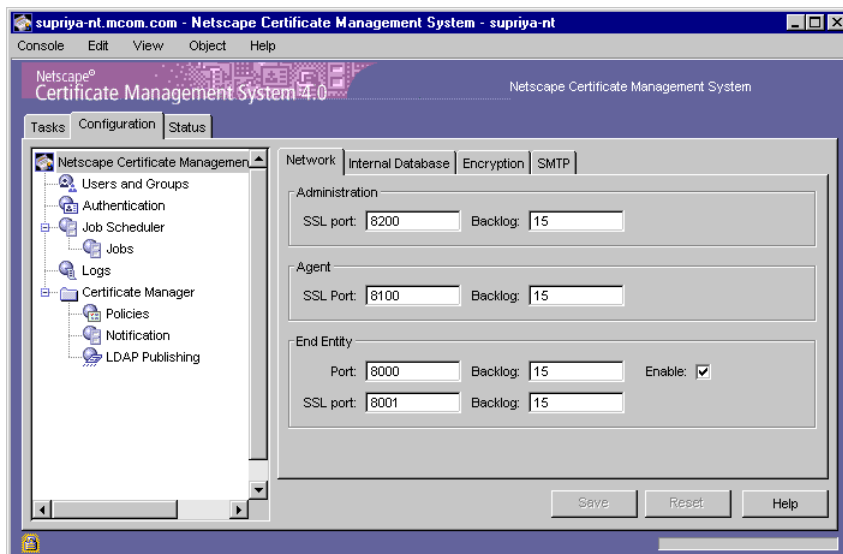
Similar to the HTTP port, you can enable or disable the HTTPS port. For example, if you don't want end-entity interaction with a Certificate Manager, you can disable the HTTPS port. For details, see “Configuring End-Entity Interaction with Subsystems” on page 537.

## Configuring Port Numbers

To change the administration, agent, or end-entity port numbers used by a CMS instance:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.

The Network tab appears.



3. To change the administration port number, enter the port number in the Administration section:

**SSL port.** Type a TCP/IP port number. Certificate Management System uses this port for SSL-enabled communications with the CMS window—that is, HTTPS requests from administrators. Make sure the port number you specify is unique on the host system.

**Backlog.** Type the number of connections that can be waiting to be serviced at the administration port. The default number is 15. The number you enter in this field is passed to the operating system's `listen()` call.

4. To change the agent port number, enter the port number in the Agent section:

**SSL port.** Type a TCP/IP port number. Certificate Management System uses this port for SSL-enabled communications with the Agent Services interface—that is, HTTPS requests from agents. Make sure the port number you specify is unique on the host system.

**Backlog.** Type the number of connections that can be waiting to be serviced at the agent port. The default number is 15. The number you enter in this field is passed to the operating system's `listen()` call.

5. To change the end-entity port numbers, enter the port numbers in the End Entity section.

Certificate Management System is capable of simultaneous SSL and non-SSL communications at the end-entity port. This means that you do not have to choose between SSL and non-SSL communications; you can use both at the same time. But if you prefer, you can disable the non-SSL port by unchecking the “Enable” option.

**Port.** Type a TCP/IP port number that is unique on the host system. Certificate Management System uses this port for non-SSL communications with the end entity services interface.

This port is provided to allow enrollments of end entities that do not support SSL; for example, HTTP requests from end entities such as routers. You can use the Enable check box to turn this port on or off. Keep in mind that if this port is enabled, end entities will be able to enroll over HTTP too, which means their certificate requests could be intercepted and replayed to the server.

**Backlog.** Type the number of connections that can be waiting to be serviced at the end entity HTTP port. The default number is 15. The number you enter in this field is passed to the operating system's `listen()` call.

**Enable.** This check box allows you to enable or disable the HTTP port. Uncheck the option if you want to disable the port.

For issuing certificates to routers (using the CEP protocol), the port must be enabled; see “Certificate Issuance to Routers” on page 613.

**SSL port.** Type a TCP/IP port number. Certificate Management System uses this port for SSL-enabled communications with the end entity services interface (that is, HTTPS requests from end entities during certificate enrollment, renewal, and revocation). Make sure the port number you specify is unique on the host system.

If you don't want end-entity interaction with a subsystem, for example, if you don't want end entities to interact with a Certificate Manager, you can disable this port too (in addition to the HTTP port). See "Configuring End-Entity Interaction with Subsystems" on page 537.

**Backlog.** Type the number of connections that can be waiting to be serviced at the end-entity HTTPS port. The default number is 15. The number you enter in this field is passed to the operating system's `listen()` call.

6. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

## Specifying IP Addresses for CMS Instances

You can configure CMS instances to listen to specific IP addresses. For example, you can install the Certificate Manager and Data Recovery Manager on a single host, in separate instances, and then configure the instances so that the Certificate Manager is served on one IP address and the Data Recovery Manager is served on another address.

To clarify this further, consider the machine that hosts the Certificate Manager and Data Recovery Manager has two Ethernet cards that respond to the IP addresses 197.1.137.97 and 197.1.137.98. You can set up the Certificate Manager to listen to port 443 for the IP address 197.1.137.97 and the Data Recovery Manager to listen to port 443 for the IP address 197.1.137.98.

To configure a CMS instance to listen to specific IP addresses:

1. Stop the CMS instance; see “Stopping Certificate Management System” on page 110.
2. Open the configuration file in a text editor; to locate the file, see “Locating the Configuration File” on page 71.
3. Add any of the following configuration parameters to the file:
  - For remote administration port, add `radm.https.listenaddr=.`
  - For agent port, add `agentGateway.https.listenaddr=.`
  - For end-entity HTTPS port, add `eeGateway.https.listenaddr=.`
  - For end-entity HTTP port, add `eeGateway.http.listenaddr=.`
4. Add the IP address as the value for the parameter you added. For example, after you enter the value, the parameter would look similar to this:  
  
`radm.https.listenaddr=197.1.137.97`
5. If necessary, repeat steps 2 and 3 for the other ports.
6. Save your changes, and close the configuration file.
7. Start the CMS instance; see “Starting Certificate Management System” on page 106.

## Internal Database

Certificate Management System performs various certificate and key-management functions in response to the requests it receives. These functions include the following:

- Storing and retrieving of certificate issuance requests
- Storing and retrieving of certificate records
- Storing of CRLs
- Storing and retrieving of end users’ encryption private key records



To fulfill these functions, Certificate Management System maintains a persistent store—a preconfigured Netscape Directory Server—referred to as the *internal database* or *local database*. The internal database is installed automatically as a part of the CMS installation. It is used as an embedded database exclusively by Certificate Management System.

The Directory Server instance used for the internal database is different from the LDAP-compliant directory that you use to manage your corporatwide data (users and groups, their certificates, CRLs, and so on). In Netscape Console, you can distinguish an internal database instance from other Directory Server instances. It is in this form:

```
slapd-<cms_instance_id>-db
```

`<cms_instance_id>` is the ID of the CMS instance that is using the database. You first specified this when you installed this server.

Keep in mind that the subsystems use the database for storing different objects. A Certificate Manager stores all the data, certificate issuance requests, certificates, CRLs, and related information; a Registration Manager only stores the certificate issuance requests it receives; and a Data Recovery Manager only stores key records and related data.

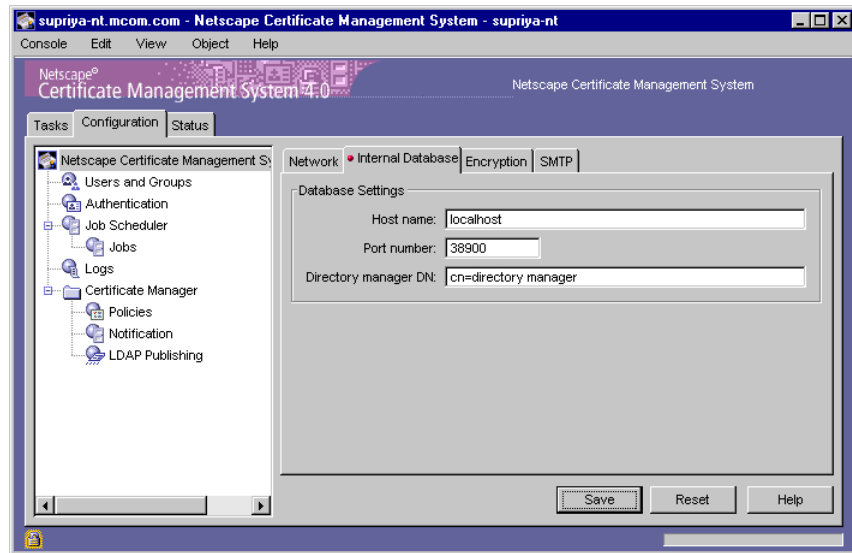
## Configuring the Internal Database

Each instance of Certificate Management System uses a Netscape Directory Server instance as its internal database. All the subsystems that were installed in a CMS instance use the same Directory Server instance to store their data. For example, if you installed a Certificate Manager and Data Recovery Manager together, they use the same internal database for data storage.

**Caution** The internal database schema is preconfigured for storing CMS data only. Do not make any changes to it or configure Certificate Management System to use any other LDAP directory. Doing so can result in loss of data. Also, do not attempt to use this database for any other purpose.

To identify the Directory Server instance that a CMS instance should use as its internal database:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab, and then in the right pane, click the Internal Database tab.



3. Identify a Directory Server instance by providing the following details:

**Host name.** Type the full host name of the machine on which Netscape Directory Server is installed. Certificate Management System uses this name to access the directory. The format for the host name is as follows:

<machine\_name>.<your\_domain>.<domain>

By default, the host name of the Directory Server instance being used as the internal database is shown as `localhost` instead of the actual host name (for example, `certificates.netscape.com`). This is done on purpose to insulate the internal database from being visible outside the system—that is, a server on `localhost` can only be accessed from the local machine. Thus, the default configuration minimizes the risk of someone connecting to this Directory Server instance from outside the local machine.

You can configure the host name to something other than `localhost` if you know what you are doing and you think you can limit the visibility of the internal database to a local subnet. For example, if you installed Certificate Management System and Directory Server on separate machines for load balancing, you will have to specify the host name of the machine in which Directory Server is installed.

**Port number.** Type a TCP/IP port number; Certificate Management System uses this port for non-SSL communications with the Directory Server instance that is functioning as the internal database. Make sure that the port you specify is unique on the host system.

**Directory manager DN.** Type the distinguished name (DN) of an entry in your LDAP directory that has read and write permission to the entire directory tree. Certificate Management System will use this DN when it accesses the directory tree to communicate with the directory. Keep in mind that the access control set up for this DN determines whether Certificate Management System can communicate with the directory. Typically, you would want to enter the directory manager's DN (the *root DN*) because this DN will have read/write permission to the entire directory tree; see “Root Distinguished Name” on page 661.

4. To save your changes, click Save.

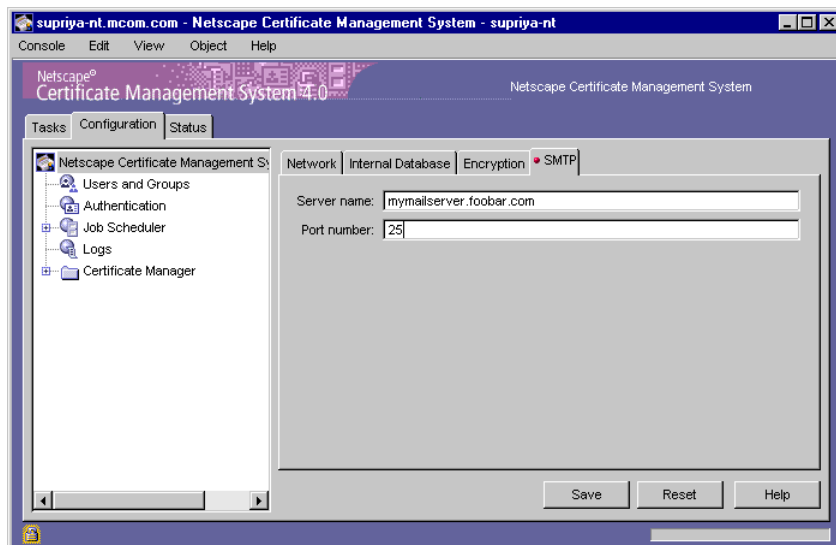
The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

## SMTP Settings

Certificate Management System can send email notifications automatically to users or agents when interesting events occur. For example, you can configure the server to send users email notifications of timed events, such as the expiration of their certificates; for details, see “Job Scheduling and Notification” on page 343.

To identify the mail server that a CMS instance should use for routing email notifications:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab, and then in the right pane, click the SMTP tab.



3. Identify the mail server by providing the following details:

**Server name.** Type the full host name of the machine on which your mail server is installed. Certificate Management System uses this name to access the mail server. The format for the host name is as follows:

`<machine_name>.<your_domain>.<domain>`

By default, the host name of the mail server is shown as `localhost` instead of the actual host name (for example, `mail.netscape.com`).

**Port number.** Type the port number at which the mail server is listening for requests.

4. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

# Managing Privileged Users and Groups

*Privileged users* are users who are designated to perform privileged operations on Netscape Certificate Management System (CMS); these operations are privileged because no one else can perform them. You assign *privileged-user* status to a user by storing the user's login information in the internal database of Certificate Management System, associating the user's login information with a personal certificate (if the user is an agent or a trusted manager), and granting access permissions to various CMS resources by adding the user to appropriate groups.

This chapter describes the types of privileged users you need to set up for a CMS instance, what each user does, how Certificate Management System identifies these users, and how you create and manage these users. The chapter also describes what a group is and discusses the groups that Certificate Management System provides by default.

The chapter has the following sections:

- Privileged-User Types and Responsibilities (page 134)
- Groups and Their Privileges (page 146)
- Setting Up Privileged Users (page 149)
- Changing Privileged-User Information (page 175)
- Deleting a Privileged User (page 181)

# Privileged-User Types and Responsibilities

After you install Certificate Management System, your first task is to set up privileged users. There are three types of privileged users: administrators, agents, and trusted managers.

- *Administrators* are users (people) who manage server-specific tasks for the Certificate Manager, Registration Manager, and Data Recovery Manager. For details, see “Administrators” on page 134.
- *Agents* are users (people) who manage the request queues for the Certificate Manager, Registration Manager, and Data Recovery Manager. For details, see “Agents” on page 135.
- *Trusted managers* are CMS subsystems that are connected to other subsystems and that are trusted to perform certain activities for them. For example, you might set up a Registration Manager to screen end-entity certificate requests for a Certificate Manager. Because the Certificate Manager trusts the Registration Manager, it approves all certificate requests received from this Registration Manager. For details, see “Trusted Managers” on page 141.

The role of a privileged user—whether administrator, agent, or trusted manager—is determined by the group to which the user belongs. This is explained in “Groups and Their Privileges” on page 146.

## Administrators

Administrators are users who have been assigned CMS administration privileges—permission to access the CMS window and perform all the system administration tasks defined there. You assign these privileges to users by adding them to the internal database and assigning membership in a group called `Administrators` that Certificate Management System creates during installation.

For each CMS instance, the server must have at least one administrator. You can also have more than one individual administering the server.

During installation, Certificate Management System prompts you to provide information for creating the first user entry in the **Administrators** group. Following installation, therefore, this group has a single user entry. For more information about this group, see “Group for Administrators” on page 146.

Certificate Management System authenticates users with administrator-level privileges based on its built-in authentication mechanism. This is explained in “Authentication of Administrators” on page 260.

## Agents

Agents are users who have been assigned end-entity certificate- and key-management privileges. Certificate Management System defines three agent roles, one for each of its subsystems: Certificate Manager agents, Registration Manager agents, and Data Recovery Manager agents.

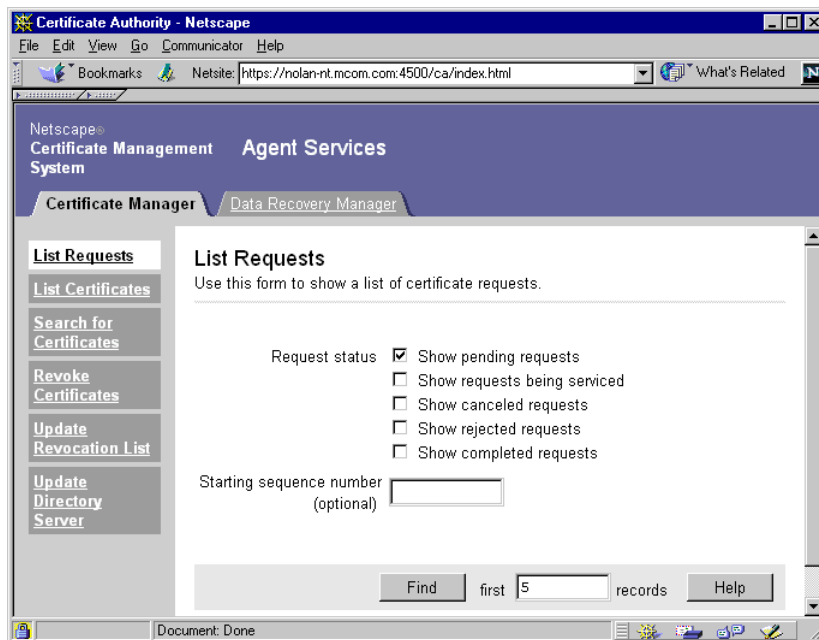
Agents interact with the Certificate Manager, Registration Manager, and Data Recovery Manager to manage operations such as these:

- List, approve, and reject pending certificate issuance and renewal requests
- List certificates
- Search for certificates
- Revoke end-entity certificates
- Manually update certificates and CRLs stored in a publishing directory
- Manage key archival and retrieval requests

All agents perform their tasks through HTML forms-based interfaces. The HTML forms an agent uses to manage a specific subsystem are grouped together and named after the subsystem they represent. For example, the forms-based interface provided for the Certificate Manager is called *Certificate Manager Agent Services* (see Figure 7.1). For more information, see “Agent Services” on page 541.

Agents cannot access the CMS window and perform the tasks provided within the Netscape Console framework—unless they are given administrator privileges.

Figure 7.1 Agents use the HTML forms-based interface called Agent Services



Each subsystem installed in a CMS instance must have at least one agent. You can also have more than one individual managing agent services.

You create agents by adding them to the internal database, assigning membership in the appropriate agent groups, and identifying certificates that the agents must use for SSL client authentication to the subsystem (for it to service requests from the agents). For information about agents' certificates, see "Agent's Certificate for SSL Client Authentication" on page 137. For information on creating agents for a CMS instance, see "Setting Up Agents" on page 152.

During installation, Certificate Management System automatically creates appropriate groups with agent privileges. For more information about these groups, see "Groups for Agents" on page 147.

Certificate Management System authenticates users with agent-level privileges based on its built-in authentication mechanism. This is explained in "Authentication of Agents" on page 262.



## Agent's Certificate for SSL Client Authentication

To make a user an agent for a subsystem, one of the things you must do is store the user's client (personal) certificate information in the internal database of the subsystem. For example, if you set up an agent for a Certificate Manager, you store the agent's client certificate in the internal database of that Certificate Manager. Then, when the subsystem receives a request from the agent, it uses this certificate to verify the authenticity of the request before servicing it. For details on how the subsystem verifies the authenticity of a request from an agent, see "Authentication of Agents" on page 262.

If the user you want to set up as an agent does not own a client certificate, ask the user to get one. Depending on your company's PKI policy, the user could get the client certificate from either an internally deployed CA or any public CA.

Keep in mind that the CA that signs your agents' certificates must be *trusted* by the subsystem that processes requests sent by these agents; for example, if your subsystems are set up not to trust public CAs, your agents should not get their certificates signed by public CAs. Make sure that the CA's certificate exists in the subsystem's trust database and that the certificate is valid and trusted. To check whether or not the CA's certificate exists in a subsystem's trust database, follow the instructions in "Viewing the Certificate Database Contents" on page 247.

- If the CA's certificate isn't listed, follow the instructions in "Using the Wizard to Install a Certificate or Certificate Chain" on page 215 and add the certificate to the subsystem's certificate database.
- If the CA's certificate is listed but *untrusted*, follow the instructions in "Changing the Trust Settings of a CA Certificate" on page 250 and change the setting to *trusted*.

## Getting an Agent's Certificate from a Public CA

The following general guidelines explain how a user can get a client certificate from a public CA and how you can copy that certificate (in base-64 encoded form) to the internal database of the appropriate subsystem:

1. The user sends a client certificate request to the public CA from the client machine that he or she will use to access the subsystem from the Agent Services interface. It is important that the user generate and submit this request from the machine she or he will use later to access the subsystem, because part of this request process generates a private key on the local

machine. Alternatively, if location independence is required, the user can use a hardware token, such as a smart card, to generate and store the key pair (and the certificate when the user receives it from the public CA).

2. When the user receives the certificate from the public CA, the user imports the certificate into the web browser that he or she will use to access the subsystem. It is a good idea to ask the user to inform you that the certificate has been installed.
3. Ask the user to send you the certificate information sent by the public CA. In the information that you receive, locate the user's certificate in base-64 encoded form.

You can also get the user's certificate from the public CA that issued it. Access the public CA site, search for the user's certificate, and locate the certificate in base-64 encoded form.

4. Copy the base-64 encoded certificate, including the -----BEGIN CERTIFICATE----- and -----END CERTIFICATE----- marker lines, to a text file.

The copied information should look similar to the following example:

```
-----BEGIN CERTIFICATE-----
MIICJzCCAaZCgAwIBAgIBAzANBgkqhkiG9w0BAQQFADBCCSAwHgYDVQQKEXdOZXRzY2FwZSBDb21td
W5pY2F0aW9uc2ngjhnMVQ2VydG1maWNhdGUGQXV0aG9yaXR5MB4XDtk4MDgyNzE5MDAwMFoXDtk5M
DIyMzE5MDAwMnBjdngngYoxIDAeBgNVBAoTF05ldHNjYXB1IENvbw11bmljYXRpb25zMQ8wDQYDVQQ
LEWZQZW9wbGUxZmFzAVBgoJkiaJkSIsZAEBEwdzdXBzaXlhMRcwFQYDVQQDEw5TdXByaXlhIFNoZXR0e
TEjMCEGCSqGS1b3DbndgJARYUc3Vwcm15YUBuZXRzY2FwZS5jb20wXDNBngkqhkiG9w0BAQEFAANL
ADBIAkEAoYiYgthgtbbnJfngjnjgnagwJjAObgNVHQ8BAf8EBAMCBLAwFAYJYIZIAyB4QgEBAQHBA
QDAgCAMA0GCSqGS1b3DQEBBAUAA4GBAFi9FzyJlLmS+kzsue0kTXawbwamGdYq12w4hIBgdR+ jWeL
mD4CP4xzmKdvQ6IqD2q8DBs9lRQu9JYg129oaCLpZfMNTPrMc3WPKO2pWZWUm7waHEtdbo9vSspbJk
XTM/2GhWbsO5vLdeOxrPGxiHkHgV/vmqCl4HW7AorqGgyfygbhgtghutrhryj
-----END CERTIFICATE-----
```

### Important

When you copy the certificate, be sure to include the -----BEGIN CERTIFICATE----- and -----END CERTIFICATE----- marker lines.

5. Save the text file and use it to store a copy of the certificate in a subsystem's internal database (see "Step 3. Store the Agent's SSL Client Certificate in the Internal Database" on page 156).

## Getting an Agent's Certificate from Certificate Management System

The following general instructions explain how a user can get a client certificate from Certificate Management System and how you can copy that certificate (in base-64 encoded form) to the internal database of a subsystem:

1. The user sends a client certificate request to Certificate Management System from the client machine that he or she will use to access the subsystem from the Agent Services interface. It is important that the user generate and submit this request from the machine he or she will use later to access the subsystem, because part of this request process generates a private key on the local machine. Alternatively, if location independence is required, the user can also use a hardware token, such as a smart card, to generate and store the key pair (and the certificate when the user receives it from the public CA).
2. Depending on how your Certificate Management System is configured for certificate issuance, one of the following events happen:
  - If Certificate Management System is configured for manual certification, an issuing agent must process the request and approve it for issuance. Once the request is approved, the server issues the client certificate to the user.
  - If Certificate Management System is configured for automated certification and the request passes authentication and policy checks, the server automatically issues the client certificate to the user.
3. When the user receives the certificate, the user must import the certificate into the web browser that he or she will use to access the subsystem. It is a good idea to ask the user to inform you that the certificate has been installed.
4. After the user imports the certificate into the web browser, you need to copy the certificate (in base-64 encoded form) in order to be able to add it to a subsystem's internal database.

To copy an agent's certificate:

1. Open a web browser window.
2. Go to the Certificate Management System home page for end entities (by default, it is called *End Entity Registration Services*).

The default URL for this page is in this form:

`http://<host_name>:<end_entity_HTTP_port>` or

`https://<host_name>:<end_entity_HTTPS_port>`

In both cases, the <host\_name> must be in this form:

`<machine_name>.<your_domain>.<domain>`

For example, the URL may look like this: `https://testCA.netscape.com`

3. Click the Retrieval tab.
4. In the left frame, click either the List Certificates or Search For Certificates link, and search for the user's certificate.
5. In the page listing the results of your search, click the Details button (next to the corresponding user's entry) to see detailed information about the certificate.
6. Scroll down to the section named "Installing This Certificate in a Client," which contains the user's certificate in base-64 encoded form.
7. Copy the base-64 encoded certificate, including the -----BEGIN CERTIFICATE----- and -----END CERTIFICATE----- marker lines, to a text file.

The copied information should look similar to the following example:

-----BEGIN CERTIFICATE-----

```
MIICJzCCAZCgAwIBAgIBAzANBgkqhkiG9w0BAQQFADBCCSAwHgYDVQQKEXdOZXRzY2FwZSBDb21td
W5pY2F0aW9uc2ngjhnMVQ2VydG1maWNhdGUgQXV0aG9yaXR5MB4XDtk4MDgyNzE5MDAwMFOXDtk5M
DIyMzE5MDAwMnBjdGngYoxIDAeBgNVBAoTF05ldHNjYXB1IENvbW11bm1jYXRpb25zMQ8wDQYDVQQ
LEWZQZW9wbGUxPzAVBgoJkiaJk1sZAEwEwdzdXByaXlhMRcwFQYDVQQDEW5TdXByaXlhIFNoZXR0e
TEjMCEGCSqGS1b3DbndgJARYUc3Vwcm15YUbuZXRzY2FwZS5jb20wXDANBgkqhkiG9w0BAQEFAANL
ADBIakeEAoYiYgthgtbbn jfng jn jgnagwJ jAOBGNVHQ8BAf8EBAMCBLAwFAYjYI ZIAYb4QgEBAQHBA
QDAgCAMA0GCSqGS1b3DQEBAUAA4GBAFi9FzyJlLmS+kz sue0kTXawbwamGdYq12w4hIBgdR+ jWeL
mD4CP4xzmKdvQ6IqD2q8DBs9lRQu9JYg129oaCLpZfMNTpNmMc3WPKO2pWZWUm7waHEt dbo9vSpbJk
XTM/2GhWbs05vLdeOxrPGxi hkgV/vmqC14HW7AorqGgyfygbhgtgutrhyrj
```

-----END CERTIFICATE-----

**Important** When you copy the certificate, be sure to include the -----BEGIN CERTIFICATE----- and -----END CERTIFICATE----- marker lines.

8. Save the text file and use it to store a copy of the certificate in a subsystem's internal database (see "Step 3. Store the Agent's SSL Client Certificate in the Internal Database" on page 156).

## Trusted Managers

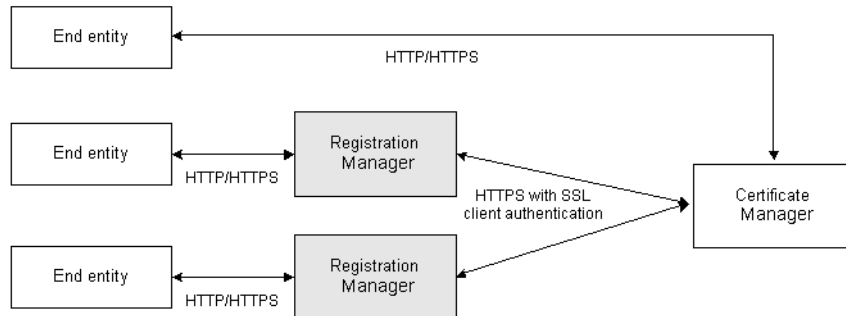
*Trusted managers* are those CMS subsystems that are connected to other CMS subsystems and that are trusted to perform specific functions for them. In other words, a trusted manager acts as a front end to the subsystem that trusts it, performing specific functions, depending on the subsystem to which it is connected. You establish this trust between the two subsystems by configuring them to function in certain way.

### Subsystems That Can Function as Trusted Managers

In Certificate Management System, the Registration Manager or Certificate Manager can function as a trusted manager; the Data Recovery Manager cannot function as a trusted manager. You can configure

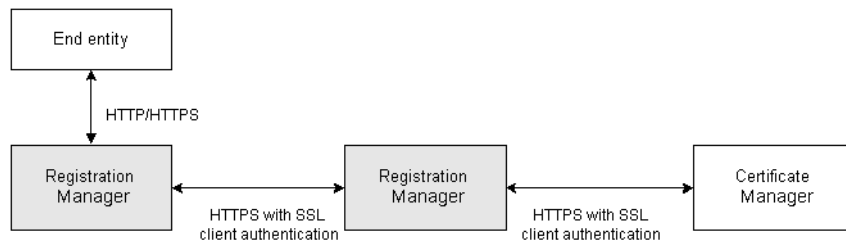
- A Certificate Manager to delegate its end-entity interactions to a trusted Registration Manager, for reasons of localizability (proximity to end entities), customizability, and CA scalability; the Certificate Manager trusts the Registration Manager and signs all certificate signing requests sent by this Registration Manager.

For example, as illustrated in the figure below, you might deploy one or more Registration Managers to process, approve, and forward certificate signing requests to a Certificate Manager.



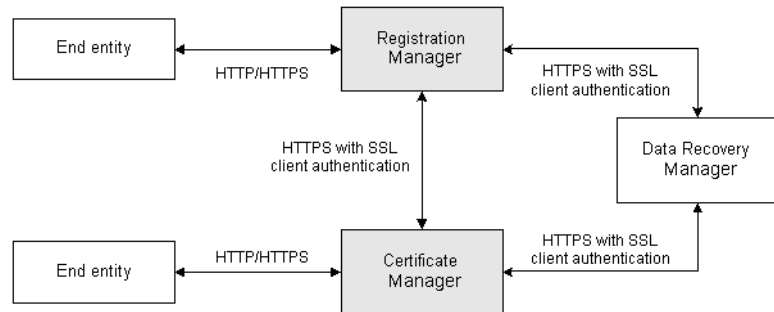
- A Registration Manager to delegate its end-entity interactions to a trusted Registration Manager, for reasons of localizability (proximity to end entities), customizability, and scalability; the Registration Manager trusts the Registration Manager and services all certificate requests sent by this Registration Manager.

For example, as illustrated in the figure below, you might deploy one or more Registration Managers to forward requests to another Registration Manager in a Registration Manager chain. (The Registration Manager passing the request acts as the client and the one receiving the request acts as the server.)



- A Data Recovery Manager to delegate its end-entity interactions to a trusted Certificate Manager or Registration Manager for security reasons; the Data Recovery Manager trusts the Certificate Manager or Registration Manager and services all key archival and recovery requests initiated by this subsystem.

For example, as illustrated in figure below, you might deploy one or more Certificate Managers or Registration Managers to send key archival or recovery requests to a Data Recovery Manager.



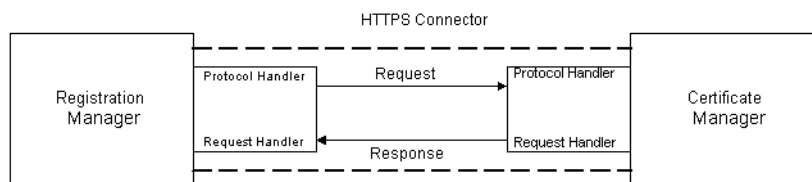
## Connectors for Linking Trusted Managers

Certificate Management System supports proprietary HTTPS connectors for linking CMS subsystems. You can use these connectors to make the following connections:

- Registration Manager to Certificate Manager
- Registration Manager to Registration Manager
- Registration Manager to Data Recovery Manager
- Certificate Manager to Data Recovery Manager

Figure 7.2 illustrates how a trusted Registration Manager communicates with a Certificate Manager.

**Figure 7.2** Connectivity service between a trusted Registration Manager and other subsystems



Keep in mind that a trusted manager does not take on the main functions of the subsystem that trusts it. For example, if a Registration Manager is connected to a Certificate Manager, the Registration Manager has no authority to issue (sign) certificates or CRLs. It receives end-entity requests, authenticates them, and forwards them to the Certificate Manager for signing. After receiving a response from the Certificate Manager, it notifies the end entity of the results.

Similarly, a Certificate Manager or Registration Manager connected to a Data Recovery Manager has no authority to archive and recover end users' encryption private keys.

You can configure a subsystem to trust one or more managers. You do this by adding these managers as privileged users to the internal database of that subsystem, assigning them memberships in the appropriate group, and identifying the certificates the managers must use for SSL client authentication to the subsystem they report to. For information about adding a trusted manager, see "Setting Up Trusted Managers" on page 158.

During installation, Certificate Management System automatically creates a group with trusted manager privileges. For more information about this group, see "Group for Trusted Managers" on page 149.

## Trusted Manager's Certificate for SSL Client Authentication

By default, a Registration Manager that has been set up to function as a trusted manager uses its *signing certificate* for SSL client authentication to the subsystem that trusts it. For information on this certificate, see "Signing Key Pair and Certificate" on page 188. Similarly, a Certificate Manager that has been set



up to function as a trusted manager uses its *SSL server certificate* for SSL client authentication to the subsystem that trusts it. For information on this certificate, see “SSL Server Key Pair and Certificate” on page 186.

When you set up a trusted manager for a CMS subsystem, it is important to know which CA has issued the certificate the trusted manager will use for SSL client authentication to the subsystem. The certificate must be issued by a CA that the subsystem trusts. For example, when you set up a trusted Registration Manager for a subsystem, it is important to know which CA has issued the Registration Manager's signing certificate. The certificate must be issued by a CA that the subsystem trusts. If the subsystem is a Certificate Manager, the certificate must be issued by either the Certificate Manager itself or a CA that the Certificate Manager trusts. Similarly, if the Registration Manager is connected to a Data Recovery Manager, the signing certificate must be issued by the CA that the Data Recovery Manager trusts.

The issuer of a Registration Manager's signing certificate is the CA from which you requested the certificate when you installed the Registration Manager. If you have renewed the certificate since installation, the issuer is the CA from which you requested the renewed certificate. Check the signing certificate for its issuer's name; see “Viewing the Certificate Database Contents” on page 247. You can also find this information by looking at the installation worksheet you completed in preparation for installing the system.

Once you learn the issuer's name, verify that this CA's certificate exists in the subsystem's trust database and that the certificate is trusted. To check whether the CA's certificate exists in the subsystem's trust database, follow the instructions in “Viewing the Certificate Database Contents” on page 247.

- If the CA's certificate isn't listed, follow the instructions in “Using the Wizard to Install a Certificate or Certificate Chain” on page 215 and add the certificate to the subsystem's certificate database.
- If the CA's certificate is listed but *untrusted*, follow the instructions in “Changing the Trust Settings of a CA Certificate” on page 250 and change the trust setting to *trusted*.

# Groups and Their Privileges

In Certificate Management System, a *group* refers to a collection of privileged users—administrators, agents, or trusted Registration Managers. Each group has predetermined privileges, based on its access control. All users belonging to a group automatically inherit the privileges of that group.

When you installed Certificate Management System, it automatically created the following groups for the subsystems you installed:

- Group for Administrators
- Groups for Agents
- Group for Trusted Managers

These default groups are created in the internal database of the appropriate CMS instance. They can help you set up your privileged users quickly and easily.

Do not delete or change the group names. Also, don't change the internal database in which the groups are stored. However, you can add new privileged users to these groups; see “Setting Up Privileged Users” on page 149.

## Group for Administrators

During installation, Certificate Management System automatically creates a group called `Administrators` and adds a user to this group; the server sets the name of this user to the *certificate administrator ID* (of the CMS administrator) you specified during installation. If you don't remember this name, see the installation worksheet you completed in preparation for installing the system. For example, if you specified `admin` as the user ID for the CMS administrator, the name of the user in the `Administrators` group will be `admin`.

Keep in mind that the `Administrators` group must always contain at least one user entry. This means you can delete the entry that was created in this group during installation, provided you add another user to the group.

After installation, you must do the following:

1. Log in to the CMS window with the administrator ID and password specified during installation.
2. Depending on the components you installed, create one or more privileged users and add them to the appropriate groups. It is recommended that you add at least one more user to the `Administrators` group. For instructions on creating privileged users and adding them to one or more groups, see “Setting Up Privileged Users” on page 149.

## Groups for Agents

Depending on the subsystems you chose to install, Certificate Management System automatically creates a combination of the following groups in a CMS instance:

- `Certificate Manager Agents` group, if you have installed the Certificate Manager
- `Registration Manager Agents` group, if you have installed the Registration Manager
- `Data Recovery Manager Agents` group, if you have installed the Data Recovery Manager

### Group for Certificate Manager Agents

When the Certificate Manager is installed, a group called `Certificate Manager Agents` is automatically created in its internal database. After installation, this group has a single user entry—when you get the first agent certificate from the Certificate Manager, the server automatically adds the initial administrator as the agent and stores a copy of the agent certificate against that user entry. The user ID for this agent user is the same as the *certificate administrator ID* (as specified during installation).

The `Certificate Manager Agents` group has access rights to agent-specific resources of the Certificate Manager; that is, privileged users you add to this group automatically inherit access rights to the agent port of the Certificate Manager. For information on ports, see “CMS Ports” on page 121.

After installation, you should add to this group the privileged users to whom you want to assign Certificate Manager agent privileges. All agents who belong to the `Certificate Manager Agents` group can access the Certificate Manager Agent Services interface; see “Certificate Manager Agent Services” on page 541.

For an agent to be able to carry on SSL client-authenticated communication with a Certificate Manager, you need to do additional configurations. See “Setting Up Agents” on page 152.

## **Group for Registration Manager Agents**

When the Registration Manager is installed, a group called `Registration Manager Agents` is automatically created in its internal database. By default, this group has no entries.

The `Registration Manager Agents` group has access rights to agent-specific resources of the Registration Manager; that is, privileged users you add to this group automatically inherit access rights to the agent ports of the Registration Manager. For information on ports, see “CMS Ports” on page 121.

After installation, you should add to this group the privileged users to whom you want to assign Registration Manager agent privileges. All agents who belong to the `Registration Manager Agents` group can access the Registration Manager Agent Services interface; see “Registration Manager Agent Services” on page 542.

For an agent to be able to do SSL client-authenticated communication with a Registration Manager, you need to do additional configurations. See “Setting Up Agents” on page 152.

## **Group for Data Recovery Manager Agents**

When the Data Recovery Manager is installed, a group called `Data Recovery Manager Agents` is automatically created in its internal database. By default, this group has no entries.

The `Data Recovery Manager Agents` group has access rights to agent-specific resources of the Data Recovery Manager; that is, privileged users you add to this group automatically inherit access rights to the agent ports of the Data Recovery Manager. For information on ports, see “CMS Ports” on page 121.

After installation, you should add to this group the privileged users to whom you want to assign Data Recovery Manager agent privileges. All agents who belong to the `Data Recovery Manager Agents` group can access the Data Recovery Manager Agent Services interface; see “Data Recovery Manager Agent Services” on page 543.

For an agent to be able to carry on SSL client-authenticated communication with a Data Recovery Manager, you need to do additional configurations. See “Setting Up Agents” on page 152.

## Group for Trusted Managers

When the Certificate Manager, Registration Manager, or Data Recovery Manager is installed, a group called `Trusted Managers` is automatically created in its internal database. By default, this group has no entries.

The `Trusted Managers` group has access rights to the corresponding agent gateway; that is, the subsystems you add to this group automatically inherit access rights to the agent port of the corresponding Certificate Manager, Registration Manager, or Data Recovery Manager. For information on ports, see “CMS Ports” on page 121.

After installation, you should add to this group the subsystems that you want to function as trusted managers. All subsystems that belong to the `Trusted Managers` group can carry on SSL client-authenticated communication with the subsystem that trusts them and receive responses back.

For a Registration Manager to be able to do SSL client-authenticated communication with a subsystem, you need to do additional configurations. See “Setting Up Trusted Managers” on page 158.

## Setting Up Privileged Users

Setting up privileged users for a CMS instance involves adding the appropriate user information to the internal database of that instance. You can set up any number of privileged users for a CMS instance. If the user is a person (that is, an administrator or agent), you can put that user into as many groups as you like.

This section describes the following tasks:

- Setting Up Administrators
- Setting Up Agents
- Setting Up Trusted Managers

## Setting Up Administrators

You need at least one administrator for each instance of Certificate Management System. To understand the role of an administrator, see “Administrators” on page 134.

This section explains how to add administrators to a CMS instance manually. To import users with administrator-level privileges from a Netscape Certificate Server version 1.0x database into the internal database used by a CMS instance, see "Appendix A, Migrating from Certificate Server" in the *Netscape Certificate Management System Deployment and Installation Guide*.

Setting up an administrator involves the following steps:

- Step 1. Find the Required Information
- Step 2. Add the Information to the Internal Database

### Step 1. Find the Required Information

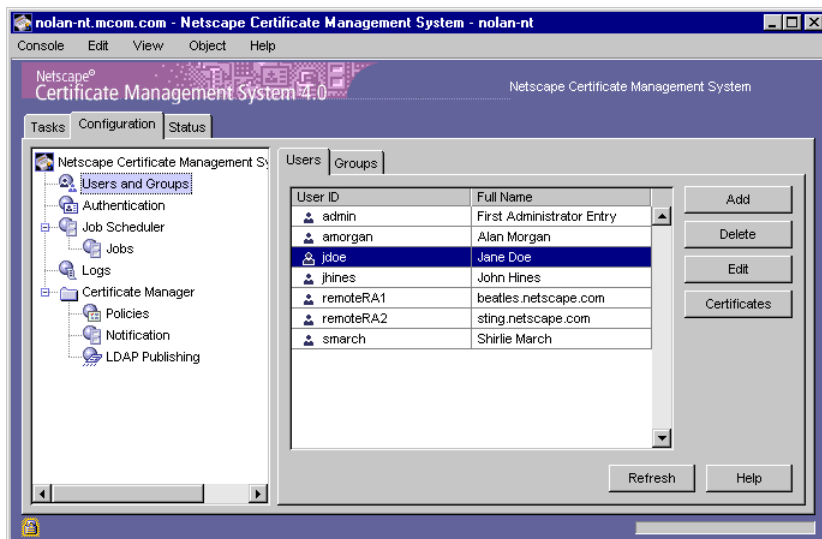
Note the user's corporate information, such as name, email address, and phone number.

## Step 2. Add the Information to the Internal Database

To add the information to the internal database of a CMS instance:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click Users and Groups.

The Users tab appears.



3. Click Add.

The Edit User Information window appears.

User ID:   
 Full name:   
 Password:   
 Confirm Password:   
 E-Mail:   
 Phone:   
 Group:

4. Specify information as appropriate:

**User ID.** Type a user ID or login name for the user. The ID can be an alphanumeric string of up to 255 characters. Give this ID to the user. The user is required to enter this ID in the login screen of the CMS window; see “Accessing the CMS Window” on page 63.

**Full name.** Type the user’s full name. The user never sees this. This field is to help you keep track of your users. The name can be an alphanumeric string of up to 255 characters.

**Password.** Type a password of up to eight characters for the user. Give this password to the user. The user is required to enter this password in the login screen of the CMS window; see “Accessing the CMS Window” on page 63.

**Confirm password.** Retype the password exactly as you typed it in the Password field.

**Email.** Type the user’s complete email address. The user never sees this. This field is to help you contact the user, if the need arises.

**Phone.** Type the user’s phone number. The user never sees this. This field is to help you contact the user, if the need arises.

**Group.** Select Administrators; for more information about this group, see “Group for Administrators” on page 146. When you set up a user, you can add him or her to only one group. To add the user to another group, see “Changing Members in a Group” on page 179.

5. Click OK.

You are returned to the Users tab. The administrator you just added will be displayed in the list of users.

6. Click Refresh to view the updated configuration.

## Setting Up Agents

You need an agent for each subsystem installed in a given CMS instance. To understand the role of an agent, see “Agents” on page 135. This section explains how to add agents to a CMS instance manually. To import users with



agent-level privileges from a Netscape Certificate Server version 1.0x database, see "Appendix A, Migrating from Certificate Server" in the *Netscape Certificate Management System Deployment and Installation Guide*.

Setting up an agent involves the following steps:

- Step 1. Find the Required Information
- Step 2. Add the Information to the Internal Database
- Step 3. Store the Agent's SSL Client Certificate in the Internal Database
- Step 4. Check the Certificate Database for the CA Certificate

## Step 1. Find the Required Information

Before adding an agent to the internal database of a CMS instance:

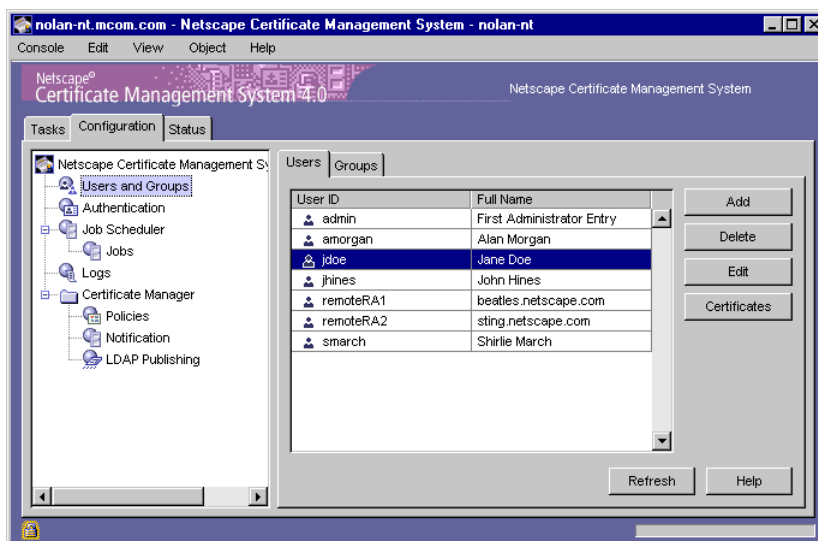
- Note the user's corporate information, such as name, login ID, password, email address, and phone number.
- Make sure the user has one or more client certificates that are currently valid; the certificate must not have expired, been revoked, or been signed by an *untrusted* authority. If the user does not own a client certificate, either issue the user a certificate or ask the user to get a certificate. For details, see "Agent's Certificate for SSL Client Authentication" on page 137.
- Identify the certificate that the user must use for SSL client authentication to Certificate Management System. You can identify more than one certificate if you want.
- Copy this certificate, in base-64 encoded format, to a text file.

## Step 2. Add the Information to the Internal Database

To add the information to the internal database of a CMS instance:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click Users and Groups.

The Users tab appears.



3. Click Add.

The Edit User Information window appears.



4. Specify information as appropriate.

The information you enter here is to help you keep track of your agent users; the user never sees or uses it. The server relies solely on the agent's client certificate (which you will add next) for authentication.

**User ID.** Type the user ID or login name. The ID can be an alphanumeric string of up to 255 characters.

**Full name.** Type the user's full name. The name can be an alphanumeric string of up to 255 characters.

**Password.** You can choose to leave this field blank.

**Confirm password.** You can choose to leave this field blank.

**Email.** Type the user's complete email address.

**Phone.** Type the user's phone number.

**Group.** Choose the appropriate agent group; for more information about this group, see "Groups for Agents" on page 147. When you set up a user, you can add her or him to only one group. To add the user to another group, see "Changing Members in a Group" on page 179.

5. Click OK.

You are returned to the Users tab. The agent you just added will be displayed in the list of users.

What you do next depends on whether you have the agent's certificate:

- If you copied the user's certificate in base-64 encoded form to a text file, proceed to Step 3. (For details on getting the user's certificate, see "Agent's Certificate for SSL Client Authentication" on page 137.)
- Otherwise, save your changes.

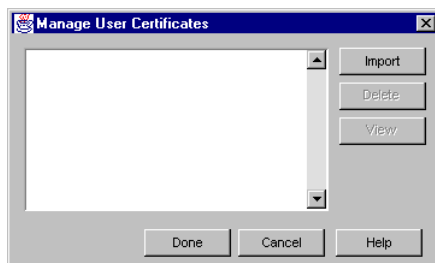
You can add the certificate to the internal database later, following the instructions provided in "Changing a Privileged User's Certificate" on page 178.

### Step 3. Store the Agent's SSL Client Certificate in the Internal Database

To store a copy of an agent's SSL client certificate in the internal database:

1. In the Users tab, click Certificates.

The Manage User Certificates window appears.

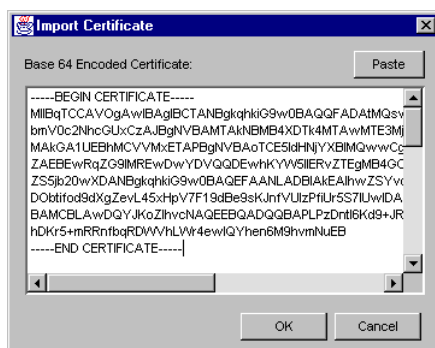


2. Click Import.

The Import Certificate window appears.

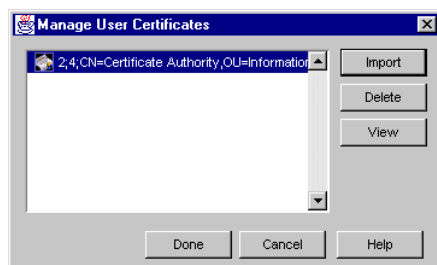
3. Click inside the text area, and paste the user's certificate in base-64 encoded form.

Be sure to include the -----BEGIN CERTIFICATE----- and -----END CERTIFICATE----- marker lines.



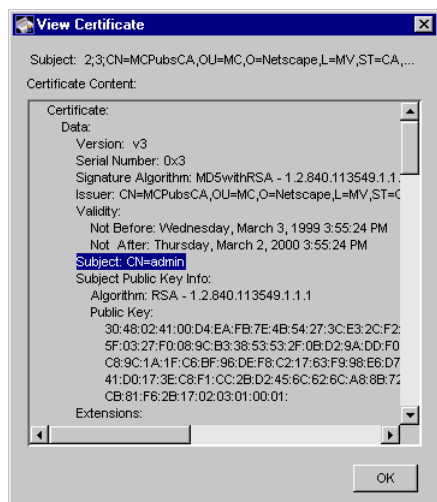
4. Click OK.

You are returned to the Manage User Certificates window. The certificate you imported should now be listed in this window.



5. To view the certificate you imported, select it and click View.

The certificate information appears.



6. Click Done.

You are returned to the Users tab.

7. Click Refresh to view the updated configuration.

## Step 4. Check the Certificate Database for the CA Certificate

The CA that signed the agent's SSL client certificate must be *trusted* by the subsystem that services requests from the agent. Make sure that this CA's certificate exists in the subsystem's certificate database (internal or external) and that it is trusted. To check whether the CA's certificate exists in your subsystem's certificate database, follow the instructions in "Viewing the Certificate Database Contents" on page 247.

- If the CA certificate isn't listed, follow the instructions in "Using the Wizard to Install a Certificate or Certificate Chain" on page 215 and add the certificate to the certificate database.
- If the CA's certificate is listed but *untrusted*, follow the instructions in "Changing the Trust Settings of a CA Certificate" on page 250 and change the trust setting to *trusted*.

## Setting Up Trusted Managers

You can set up a Registration Manager or Certificate Manager to function as a trusted manager to another CMS instance. This section explains how to do this.

- Setting Up a Registration Manager as a Trusted Manager
- Setting Up a Certificate Manager as a Trusted Manager

To understand the role of a trusted manager in your PKI, see "Trusted Managers" on page 141.

### Setting Up a Registration Manager as a Trusted Manager

You can set up a remote Registration Manager to function as a trusted manager to a Certificate Manager, Registration Manager, or Data Recovery Manager. The setup process involves the following steps:

- Step 1. Find the Required Information
- Step 2. Create a User Entry for the Registration Manager

- Step 3. Copy the Registration Manager's Certificate to the Internal Database
- Step 4. Check the Certificate Database for the CA Certificate
- Step 5. Configure the Connector Settings

## Step 1. Find the Required Information

Before setting up a Registration Manager to function as a trusted manager to another CMS subsystem:

- Note identifying information, such as the instance ID and host name of the Registration Manager.
- Make sure that the Registration Manager has the certificate you want it to use for SSL client authentication to the subsystem that will trust it; by default, the Registration Manager uses its *signing certificate* for this purpose. The certificate must be currently valid; the certificate must not have expired, been revoked, or been signed by an authority *untrusted* by the subsystem. For details, see “Trusted Manager's Certificate for SSL Client Authentication” on page 144.
- Locate the certificate in base-64 encoded format. Copy the certificate, including the -----BEGIN CERTIFICATE----- and -----END CERTIFICATE----- marker lines, to a text file.
- Identify the subsystem—Certificate Manager, Registration Manager, or Data Recovery Manager—to which you want to connect the Registration Managers. Note details, such as the host name and port number of that subsystem.
- If you are planning to connect the Registration Manager to a Certificate Manager, you should keep this in mind: during the installation of a Registration Manager, you generated a signing certificate for the Registration Manager. If you requested the signing certificate from a Certificate Manager, you were given an opportunity to add the Registration Manager as a trusted manager to that Certificate Manager's database. If you chose this option, then the Registration Manager is already set up to function as a trusted manager to that Certificate Manager—in this case, you are not required to go through these steps.

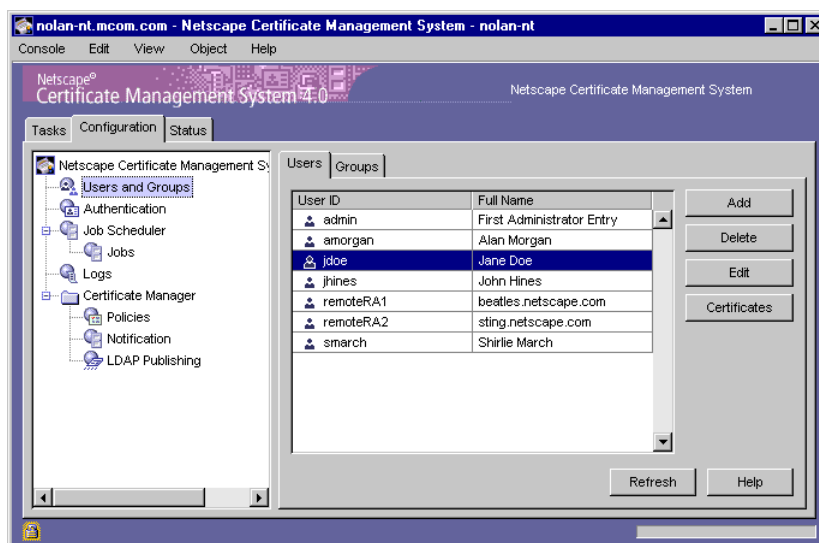
## Step 2. Create a User Entry for the Registration Manager

In this step, you create a privileged-user entry for the Registration Manager in the internal database of the subsystem. As a part of creating this entry, you also add the user entry to the `Trusted Managers` group in order to give the entry access privileges to the agent port of the subsystem.

To create a user entry with appropriate access privileges for a Registration Manager:

1. Access the CMS window for the subsystem (see “Accessing the CMS Window” on page 63).
2. Click Users and Groups.

The Users tab appears.



3. Click Add.



The Edit User Information window appears.

4. Specify information as appropriate.

The information you enter here is to help you keep track of the Registration Manager; the subsystem never uses it. The subsystem relies solely on the Registration Manager's SSL client certificate (which you will add in Step 3) for authentication.

**User ID.** Type the Registration Manager's instance ID (or any other ID that will help you identify the Registration Manager in the list of privileged users). The ID can be an alphanumeric string of up to 255 characters.

**Full name.** Type the full host name of the Registration Manager. The host name can be an alphanumeric string of up to 255 characters. It must be in this form: <machine\_name>.<your\_domain>.<domain>

**Password.** Leave this field blank.

**Confirm password.** Leave this field blank.

**Email.** Leave this field blank.

**Phone.** Leave this field blank.

**Group.** Select Trusted Managers; for more information about this group, see "Group for Trusted Managers" on page 149.

5. Click OK.

You are returned to the Users tab. The Registration Manager you just added is displayed in the list of users.

What you do next depends on whether you have the Registration Manager's SSL client certificate:

- If you copied the Registration Manager's certificate in base-64 encoded form to a text file, proceed to Step 3. (For details on getting this certificate, see "Trusted Manager's Certificate for SSL Client Authentication" on page 144.)
- Otherwise, skip to Step 5. You can add the certificate later, following the instructions in "Changing a Privileged User's Certificate" on page 178.

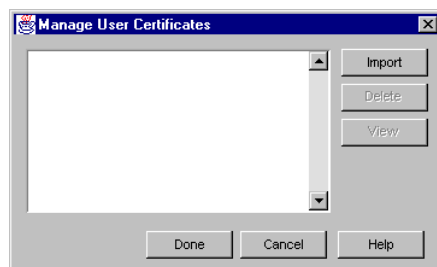
### Step 3. Copy the Registration Manager's Certificate to the Internal Database

In this step, you add a copy of the Registration Manager's SSL client authentication certificate to the internal database of the subsystem and associate the certificate with the user entry you created in Step 2.

To store the Registration Manager's SSL client certificate in the internal database of the subsystem:

1. In the Users tab, select the user entry you just added for the Registration Manager and click Certificates.

The Manage User Certificates window appears.

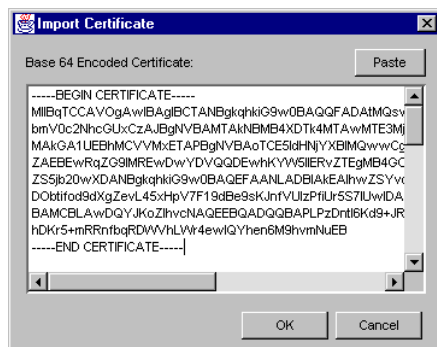


2. Click Import.

The Import Certificate window appears.

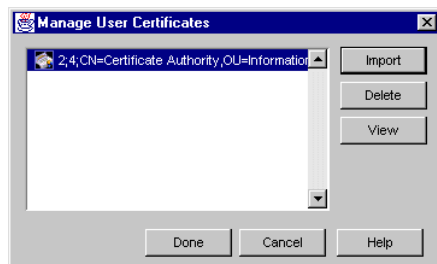
3. Click inside the text area, and paste the Registration Manager's certificate in base-64 encoded form.

Be sure to include the -----BEGIN CERTIFICATE----- and -----END CERTIFICATE----- marker lines.



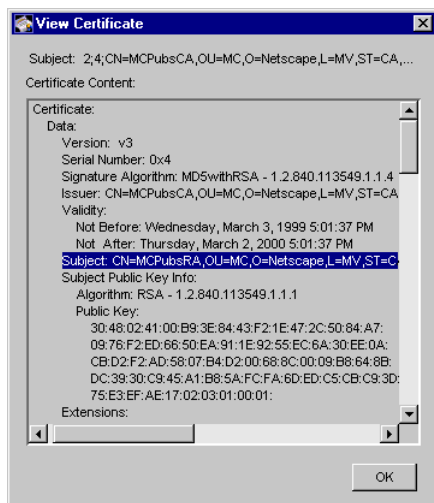
4. Click OK.

You are returned to the Manage User Certificates window. The certificate you imported should now be listed in this window.



5. To view the certificate you imported, select it and click View.

The certificate information appears. Verify that the certificate you added is the correct one.



6. Click Done.

You are returned to the Users tab.

#### Step 4. Check the Certificate Database for the CA Certificate

The issuer of the Registration Manager's certificate that you added in Step 3 must be *trusted* by the subsystem that services certificate requests approved by the Registration Manager. Make sure that this CA's certificate exists in the subsystem's certificate database (internal) and that it is trusted. To check whether the CA's certificate exists in the subsystem's certificate database, follow the instructions in "Viewing the Certificate Database Contents" on page 247.

- If the CA certificate isn't listed, follow the instructions in "Using the Wizard to Install a Certificate or Certificate Chain" on page 215 and add the certificate to the certificate database.
- If the CA's certificate is listed but *untrusted*, follow the instructions in "Changing the Trust Settings of a CA Certificate" on page 250 and change the trust setting to *trusted*.

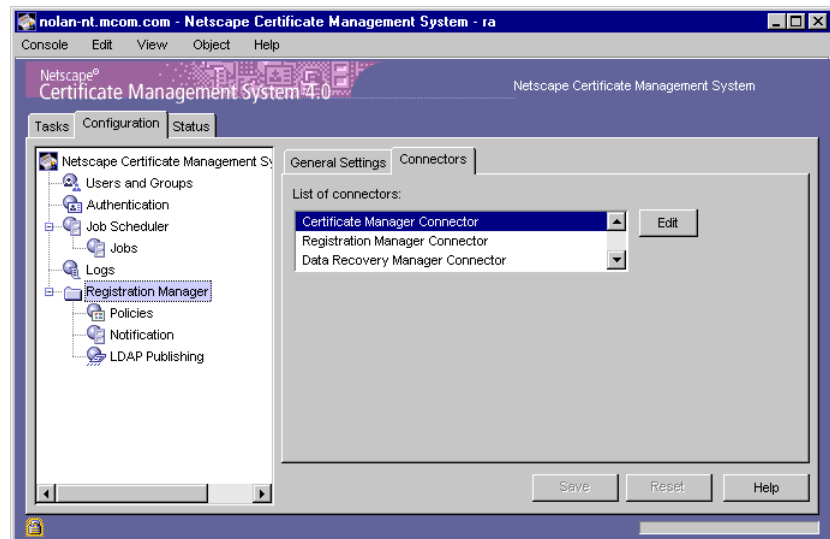
## Step 5. Configure the Connector Settings

In this step, you configure the connector settings of the Registration Manager. This enables the Registration Manager to utilize the proprietary HTTPS connectors to communicate with the subsystem (following successful SSL client authentication).

1. Access the CMS window for the Registration Manager (see “Accessing the CMS Window” on page 63).
2. In the navigation tree, click Registration Manager.

The General Settings tab appears.

3. Click the Connectors tab.

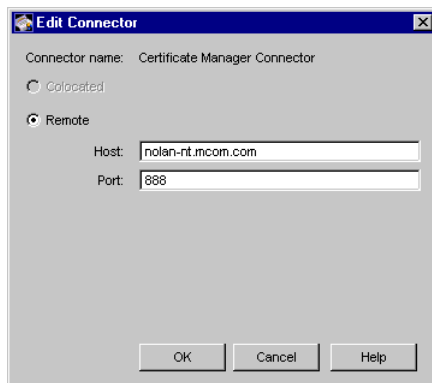


4. In the “List of connectors” select the connector:
  - If you are connecting the Registration Manager to a Certificate Manager, select Certificate Manager Connector and click Edit.
  - If you are connecting the Registration Manager to another Registration Manager, select Registration Manager Connector and click Edit.

- If you are connecting the Registration Manager to a Data Recovery Manager, select Data Recovery Manager Connector and click Edit.

For the purposes of completing these instructions, let us assume you selected Certificate Manager Connector.

The Edit Connector dialog box appears.



5. Click the Enable checkbox to enable the connector configuration.
6. Click Remote, and enter the appropriate information:

**Host name.** Type the full host name of the subsystem that trusts this Registration Manager; in this case, it would be the host name of the Certificate Manager. The Registration Manager uses this name to locate the Certificate Manager. The format for the host name must be as follows:

`<machine_name>.<your_domain>.<domain>`

**Port number.** Type the number of the TCP/IP port at which the Certificate Manager will listen to requests from the trusted Registration Manager. The default port designated for communication between a trusted Registration Manager and a subsystem is the agent's port. See "Agent Port" on page 123.

The sample screen above shows how to connect the Registration Manager to a Certificate Manager running on a host called `nolan-nt.mcom.com` listening for HTTPS requests on port 888.

7. Click OK.

You are returned to the Connectors tab.

8. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, stop and restart the server.

## Setting Up a Certificate Manager as a Trusted Manager

You can set up a Certificate Manager to function as a trusted manager to a remote Data Recovery Manager. The setup process involves the following steps:

- Step 1. Find the Required Information
- Step 2. Create a User Entry for the Certificate Manager
- Step 3. Copy the Certificate Manager's Certificate to the Internal Database
- Step 4. Check the Certificate Database for the CA Certificate
- Step 5. Configure the Connector Settings

### Step 1. Find the Required Information

Before setting up a Certificate Manager to function as a trusted manager to a Data Recovery Manager:

- Note identifying information, such as the instance ID and host name of the Certificate Manager.
- Make sure that the Certificate Manager has the certificate you want it to use for SSL client authentication to the Data Recovery Manager that will trust it; by default, the Certificate Manager uses its SSL server certificate for this purpose. The certificate must be currently valid; the certificate must not have expired, been revoked, or been signed by an authority *untrusted* by the subsystem. For details, see “Trusted Manager's Certificate for SSL Client Authentication” on page 144.
- Locate the certificate in base-64 encoded format. Copy the certificate, including the -----BEGIN CERTIFICATE----- and -----END CERTIFICATE----- marker lines, to a text file.

- Identify the Data Recovery Manager to which you want to connect the Certificate Manager. Note details, such as the host name and port number of that Data Recovery Manager.

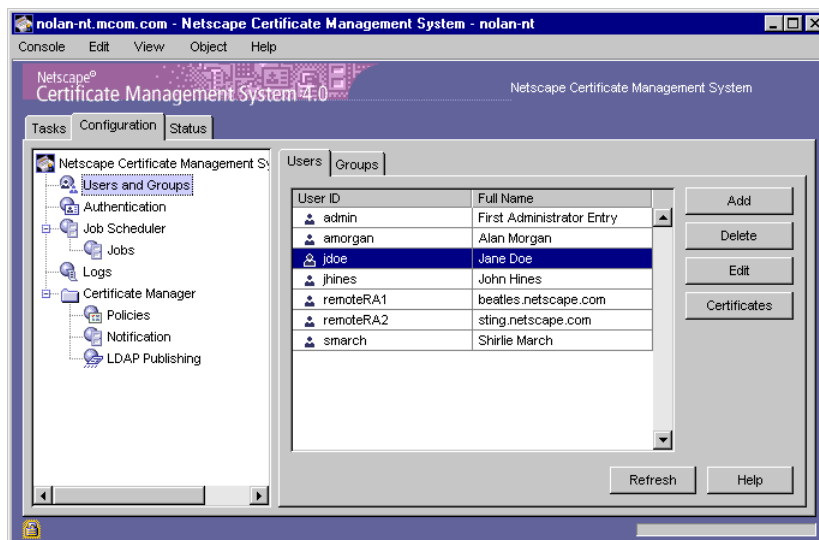
## Step 2. Create a User Entry for the Certificate Manager

In this step, you create a privileged-user entry for the Certificate Manager in the internal database of the Data Recovery Manager. As a part of creating this entry, you also add the user entry to the **Trusted Managers** group in order to give the entry access privileges to the agent port of the Data Recovery Manager.

To create a user entry with appropriate access privileges for a Certificate Manager:

1. Access the CMS window for the subsystem (see “Accessing the CMS Window” on page 63).
2. Click **Users and Groups**.

The **Users** tab appears.





3. Click Add.

The Edit User Information window appears.

4. Specify information as appropriate.

The information you enter here is to help you keep track of the Certificate Manager; the Data Recovery Manager never uses it. The Data Recovery Manager relies solely on the Certificate Manager's SSL server certificate (which you will add in Step 3) for authentication.

**User ID.** Type the Certificate Manager's instance ID (or any other ID that will help you identify the Certificate Manager in the list of privileged users). The ID can be an alphanumeric string of up to 255 characters.

**Full name.** Type the full host name of the Certificate Manager. The host name can be an alphanumeric string of up to 255 characters. It must be in this form: <machine\_name>.<your\_domain>.<domain>

**Password.** Leave this field blank.

**Confirm password.** Leave this field blank.

**Email.** Leave this field blank.

**Phone.** Leave this field blank.

**Group.** Select Trusted Managers; for more information about this group, see "Group for Trusted Managers" on page 149.

5. Click OK.

You are returned to the Users tab. The Certificate Manager you just added is displayed in the list of users.

What you do next depends on whether you have the Certificate Manager's SSL server certificate:

- If you copied the Certificate Manager's certificate in base-64 encoded form to a text file, proceed to Step 3. (For details on getting this certificate, see "Trusted Manager's Certificate for SSL Client Authentication" on page 144.)
- Otherwise, skip to Step 5. You can add the certificate later, following the instructions in "Changing a Privileged User's Certificate" on page 178.

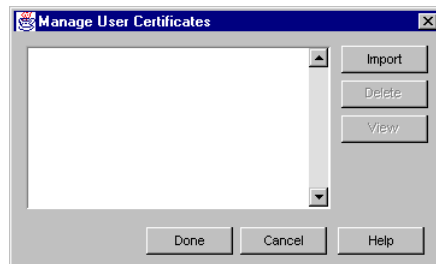
### Step 3. Copy the Certificate Manager's Certificate to the Internal Database

In this step, you add the Certificate Manager's SSL server certificate to the internal database of the Data Recovery Manager and associate the certificate with the user entry you created in Step 2.

To store the Certificate Manager's SSL server certificate in the internal database of the subsystem:

1. In the Users tab, select the user entry you just added for the Certificate Manager and click Certificates.

The Manage User Certificates window appears.

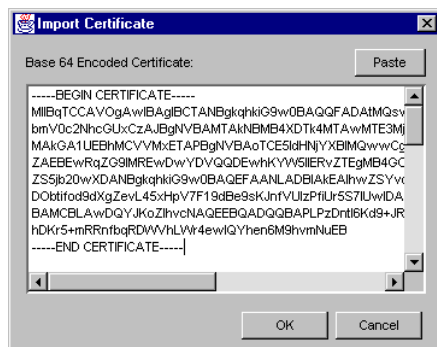


2. Click Import.

The Import Certificate window appears.

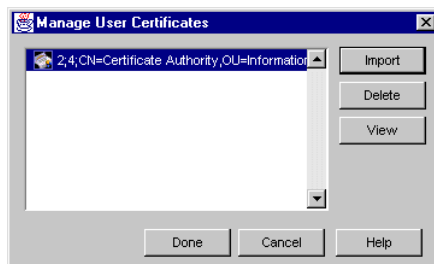
3. Click inside the text area, and paste the Certificate Manager's certificate in base-64 encoded form.

Be sure to include the -----BEGIN CERTIFICATE----- and -----END CERTIFICATE----- marker lines.



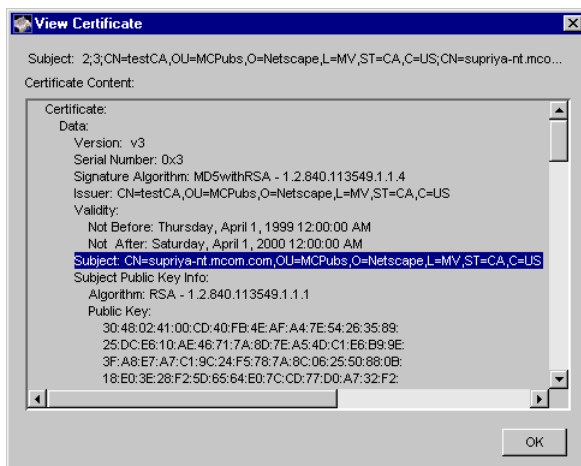
4. Click OK.

You are returned to the Manage User Certificates window. The certificate you imported should now be listed in this window.



5. To view the certificate you imported, select it and click View.

The certificate information appears. Verify that the certificate you added is the correct one.



6. Click Done.

You are returned to the Users tab.

#### Step 4. Check the Certificate Database for the CA Certificate

The issuer of the Certificate Manager's certificate that you added in Step 3 must be *trusted* by the Data Recovery Manager that services the key archival requests initiated by the Certificate Manager. Make sure that this CA's certificate exists in the Data Recovery Manager's certificate database (internal) and that it is trusted. To check whether the CA's certificate exists in the Data Recovery Manager's certificate database, follow the instructions in "Viewing the Certificate Database Contents" on page 247.

- If the CA certificate isn't listed, follow the instructions in "Using the Wizard to Install a Certificate or Certificate Chain" on page 215 and add the certificate to the certificate database.
- If the CA's certificate is listed but *untrusted*, follow the instructions in "Changing the Trust Settings of a CA Certificate" on page 250 and change the trust setting to *trusted*.

## Step 5. Configure the Connector Settings

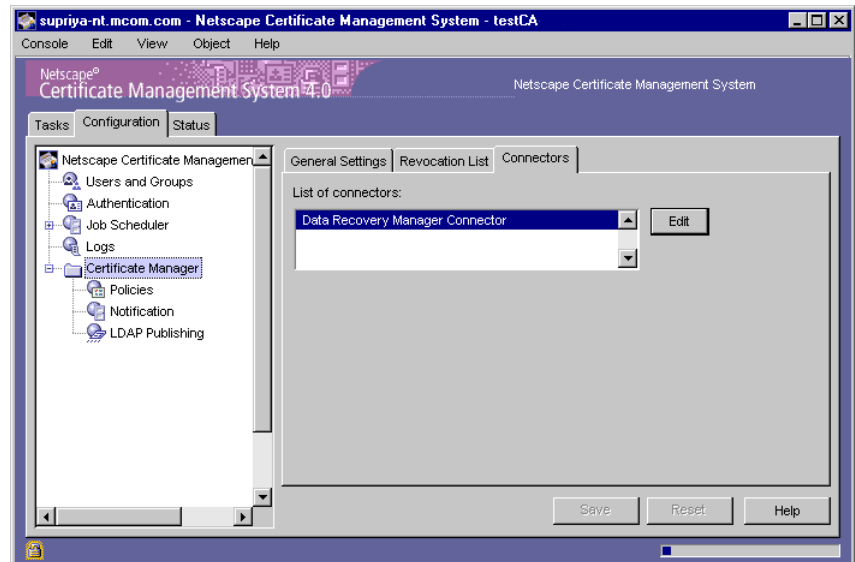
In this step you configure the connector settings of the Certificate Manager. This enables the Certificate Manager to utilize the proprietary HTTPS connectors to communicate with the Data Recovery Manager (following successful SSL client authentication).

Note that during the installation of a Data Recovery Manager, you were prompted to specify the host name and port number of the Certificate Manager to which the Data Recovery Manager will be connected. If you specified this information, you are not required to go through this step. However, it is recommended that you verify the connector setting and make sure that the information you entered during installation is correct.

1. Access the CMS window for the Certificate Manager (see “Accessing the CMS Window” on page 63).
2. In the navigation tree, click Certificate Manager.

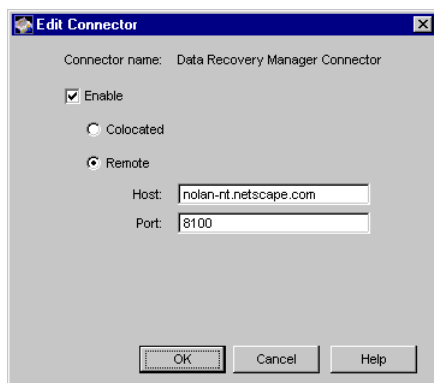
The General Settings tab appears.

3. Click the Connectors tab.



4. In the “List of connectors” select Data Recovery Manager Connector and click Edit.

The Edit Connector dialog box appears.



5. Click the Enable checkbox to enable the connector configuration.
6. Click Remote, and enter the appropriate information:

**Host name.** Type the full host name of the Data Recovery Manager that trusts this Certificate Manager. The Certificate Manager uses this name to locate the Data Recovery Manager. The format for the host name must be as follows:

`<machine_name>.<your_domain>.<domain>`

**Port number.** Type the number of the TCP/IP port at which the Data Recovery Manager will listen to requests from the trusted Certificate Manager. The port designated for communication between a trusted Certificate Manager and a Data Recovery Manager is the agent port. See “Agent Port” on page 123.

The sample screen above shows how to connect the Certificate Manager to a Data Recovery Manager running on a host called `nolan-nt.netscape.com` listening for HTTPS requests at port 8100.

7. Click OK.

You are returned to the Connectors tab.

8. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, stop and restart the server.

## Changing Privileged-User Information

You can change privileged-user information in several ways:

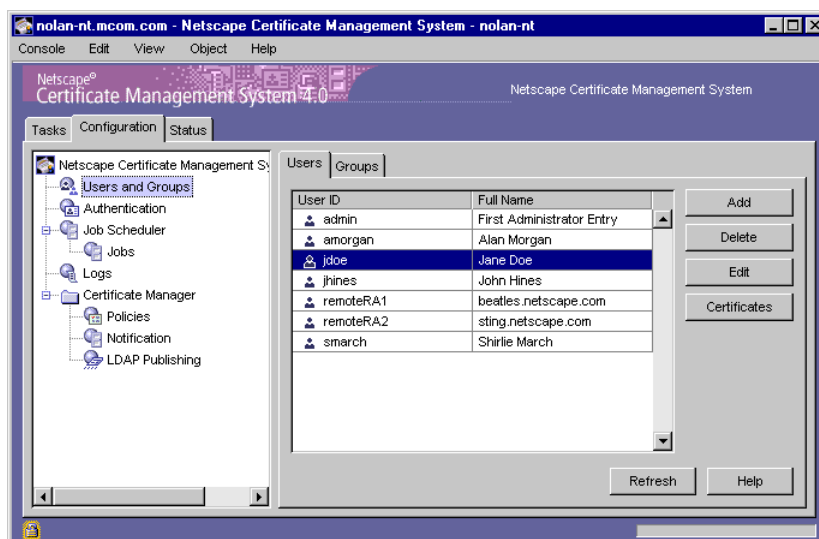
- To change the login information of a privileged user, see “Changing a Privileged User’s Login Information” on page 176.
- To add or remove certificates of a privileged user, see “Changing a Privileged User’s Certificate” on page 178.
- To change the group membership or access permissions of a privileged user, see “Changing Members in a Group” on page 179.

## Changing a Privileged User's Login Information

To change a privileged user's login information:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click Users and Groups.

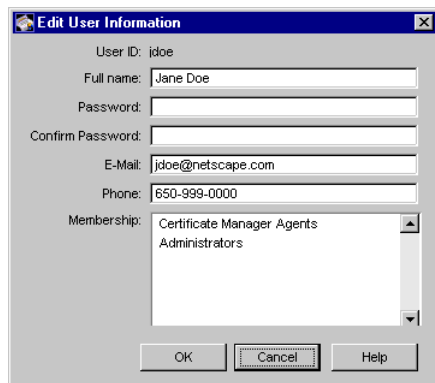
The Users tab appears.





3. In the User ID list, select the user you want to edit, and click Edit.

The Edit User Information window appears.



The screenshot shows a Windows-style dialog box titled "Edit User Information". It contains the following fields and controls:

- User ID:** jdoe
- Full name:** Jane Doe
- Password:** (empty text box)
- Confirm Password:** (empty text box)
- E-Mail:** jdoe@netscape.com
- Phone:** 650-999-0000
- Membership:** A list box containing "Certificate Manager Agents" and "Administrators".
- Buttons:** OK, Cancel, and Help at the bottom.

4. Make the appropriate modifications.

If you need details about individual fields, see "Setting Up Privileged Users" on page 149.

5. Click OK.

You are returned to the Users tab.

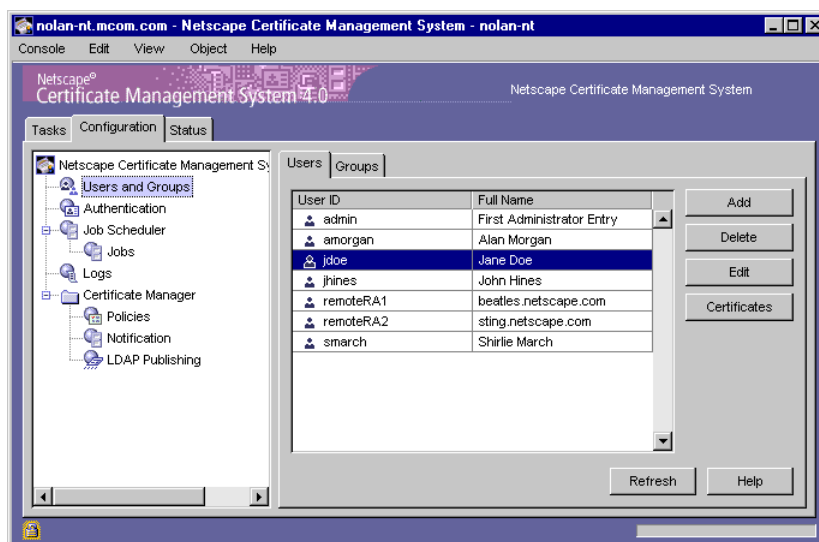
6. Click Refresh to view the updated configuration.

## Changing a Privileged User's Certificate

To change a privileged user's certificate:

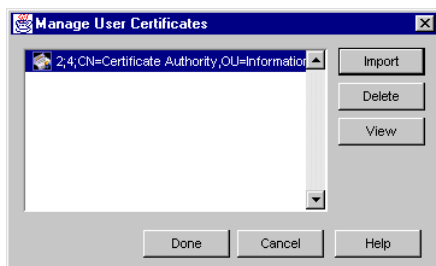
1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click Users and Groups.

The Users tab appears.



3. In the User ID list, select the user whose certificate information you want to change, and click Certificates.

The Manage User Certificate window appears.



4. Take the appropriate action:
  - To view a certificate, select the certificate and click View.
  - To delete a certificate, select the certificate and click Delete.
  - To add a new certificate for this user to the internal database, click Import. In the Import Certificate window that appears, paste the new certificate in the text area. Be sure to paste the entire base-64 encoded block, including the -----BEGIN CERTIFICATE----- and -----END CERTIFICATE----- marker lines. For details on getting the user's certificate, see "Agent's Certificate for SSL Client Authentication" on page 137.
5. Click Done.

You are returned to the Users tab.
6. Click Refresh to view the updated configuration.

## Changing Members in a Group

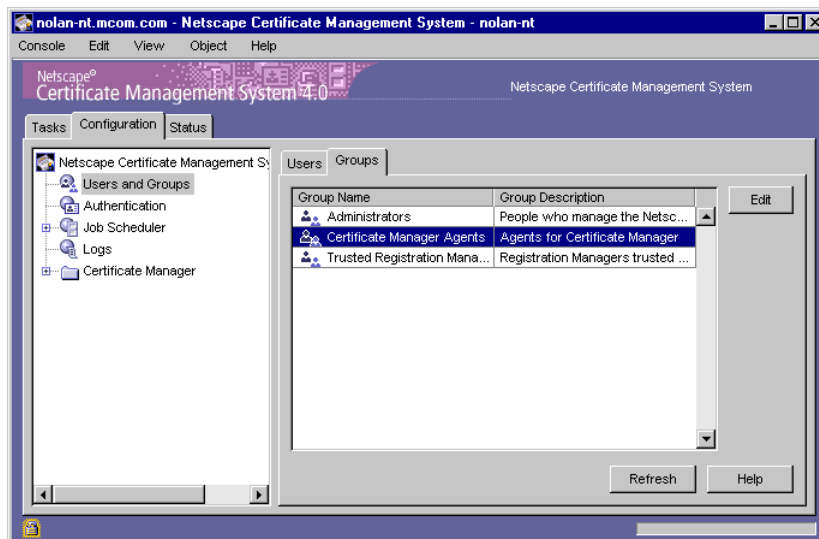
You can add or remove members from all groups. Keep in mind that the group for administrators must have at least one user entry. For details, see "Groups and Their Privileges" on page 146.

To change a group's members:

1. Access the CMS window (see "Accessing the CMS Window" on page 63).
2. Click Users and Groups.

The Users tab appears.

3. Click the Groups tab.



4. In the Group Name list, select the group you want to change, and click Edit.

The Edit Group Information window appears.



5. Make the appropriate changes:
  - To change the group description, type a new description in the "Group description" field.
  - To remove a user from the group, select the user and click Delete.

- To add users, click Add User. In the User Selection window that appears, select the users you want to add and click OK. You are returned to the Edit Group Information window.
6. Click OK when you are done with the changes.  
You are returned to the Groups tab.
  7. Click Refresh to view the updated configuration.

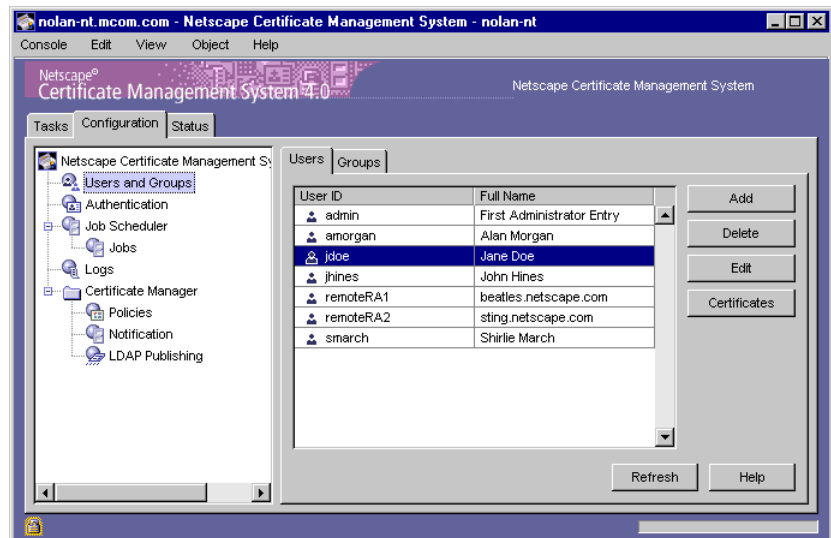
## Deleting a Privileged User

You can delete privileged users from the internal database. Before deleting a user, make sure the user is not a member of any of the groups.

To delete a privileged user from the internal database:

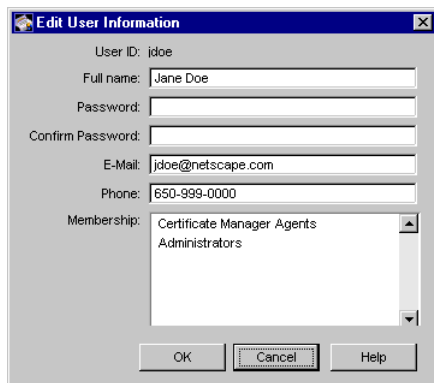
1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click Users and Groups.

The Users tab appears.



3. In the User ID list, select the user you want to delete, and click Edit.

The Edit User Information window appears.



4. Check the Membership area to see whether the user belongs to any of the groups.
  - Note down the names of any groups listed in this area.
  - Remove the user from these groups by clicking the Groups tab and following the instructions in “Changing Members in a Group” on page 179.
5. When you have removed the user from all groups, click the Users tab again.
6. Select the user again and click Delete.
7. When prompted, confirm the deletion.

The user entry is deleted from the internal database.

8. Click Refresh to view the updated configuration.

# Keys and Certificates

The main subsystems of Netscape Certificate Management System (CMS)—the Certificate Manager, Registration Manager, and Data Recovery Manager—use certificates for authentication during SSL-enabled communication. For example, when a Registration Manager forwards a certificate issuance request to a Certificate Manager for signing, the Certificate Manager expects the Registration Manager to have performed SSL client authentication before processing the request.

When you installed Certificate Management System, the installation program prompted you to generate the required certificates for the subsystems you chose to install. This chapter provides an overview of those certificates and it explains how to perform operations such as renewing the existing certificates before their validity period expires, getting new certificates for the subsystems, adding trusted CA certificates and certificate chains to the CMS trust database, and changing the trust setting of CA certificates. The chapter also explains the certificate Setup Wizard, which automates the process of requesting and installing certificates.

The chapter has the following sections:

- Keys and Certificates for the Main Subsystems (page 184)
- Tokens for Storing Keys and Certificates (page 192)
- Hardware Cryptographic Accelerators (page 198)

- Certificate Setup Wizard (page 199)
- Configuring the Server's Security Preferences (page 223)
- Getting New Certificates for the Subsystems (page 232)
- Renewing Certificates for the Subsystems (page 239)
- Managing the Certificate Database (page 246)

## Keys and Certificates for the Main Subsystems

This section explains the various certificates required and used by the Certificate Manager, Registration Manager, and Data Recovery Manager. The key pairs that correspond to certificates used by these subsystems can be stored either in an internal or an external token, or in both. It depends on the token you chose for the generation and storage of the keys and certificates. For information on tokens, see “Tokens for Storing Keys and Certificates” on page 192.

### Certificate Manager's Key Pairs and Certificates

The Certificate Manager uses the following key pairs and corresponding certificates:

- CA Signing Key Pair and Certificate
- SSL Server Key Pair and Certificate

#### CA Signing Key Pair and Certificate

Every Certificate Manager you installed has a certificate, identified as the *Certificate Manager CA signing certificate*, whose public key corresponds to the private key the Certificate Manager uses to sign the certificates it issues. The first time you generated this certificate is when you installed the Certificate



Manager. The default nickname for the certificate is `caSigningCert cert-<instance_id>`, where `<instance_id>` identifies the CMS instance in which the Certificate Manager is installed.

The subject name of the CA signing certificate reflects the name of your certificate authority (CA) as specified during the installation. All certificates signed or issued by the Certificate Manager include this name to identify the issuer of the certificate.

**Important** You cannot change the CA name; doing so would make all previously issued certificates invalid.

The Certificate Manager's status as a root or subordinate CA is determined by whether its CA signing certificate is self-signed or is signed by another CA.

- If the Certificate Manager is a root CA, its CA signing certificate is self-signed.
- If the Certificate Manager is a subordinate CA, its CA signing certificate is signed by another CA, usually the one that is a level above in the CA hierarchy (which may or may not be a root CA). If you have deployed the Certificate Manager as a subordinate CA in a CA hierarchy, you must import your root CA's signing certificate into individual clients and servers before you can use the Certificate Manager to issue certificates to them.

As an administrator, you must make sure that the private key that corresponds to the CA signing certificate is adequately protected. This includes protecting it from damage (in other words, by archiving and backing up the key) as well as protecting it from unauthorized access or use. The password that protects the token containing this key must be carefully guarded. Access to the token itself should be limited. (See "Tokens for Storing Keys and Certificates" on page 192.)

- If the key is in the internal token (the `key3.db` file), make sure that only you or authorized administrators have access to this file. It's also important to know if the file is stored on backup tapes or is otherwise available for someone to intercept. Because the destruction of a private key in a disk crash can be disastrous if you are depending upon that key for a hierarchy of certificate authorities, backing up your key data is commensurately important. If you do make copies of your keys, however, you must protect your backups with the same level of security that you use for protecting your original keys.
- If the key is in an external token, such as a smart card, keep it in a locked facility.

Also, periodically change the password that protects this key. See “Changing a Token’s Password” on page 198.

Like any other certificate, the Certificate Manager’s CA signing certificate has a validity period. You must renew the certificate before it expires. For instructions on renewing the certificate, see “Renewing Certificates for the Subsystems” on page 239.

The CA signing key pair must be well protected to ensure that it is never compromised. However, if you know or suspect that the key pair has been compromised, reissue the certificate with a new key pair. For instructions on getting a new certificate, see “Getting New Certificates for the Subsystems” on page 232.

**Important** Reissuing the Certificate Manager’s CA signing certificate with a new key pair invalidates all certificates that have been signed by the old key pair.

## SSL Server Key Pair and Certificate

Every Certificate Manager you have installed has at least one *SSL server certificate*. The first time you generated this certificate is when you installed the Certificate Manager. The default nickname for the certificate is `Server-Cert cert-<instance_id>`, where `<instance_id>` identifies the CMS instance in which the Certificate Manager is installed.

The Certificate Manager’s SSL server certificate was issued by the CA to which you submitted the certificate signing request. You might have submitted the request to the Certificate Manager itself, another internally deployed CA, or a public CA. To find out the issuer name, follow the instructions in “Viewing the Certificate Database Contents” on page 247.

The Certificate Manager uses its SSL server certificate to do SSL server-side authentication to the following:

- Netscape Console
- The End-Entity Services interface (the HTTPS port)
- The Certificate Manager Agent Services interface

By default, the Certificate Manager uses a single SSL server certificate for authentication purposes. However, you can request and install additional SSL server certificates for the Certificate Manager. For example, you can configure the Certificate Manager to use separate server certificates for authenticating to

Netscape Console, the End-Entity Services interface, and the Certificate Manager Agent Services interface. For instructions, see “Configuring the Server to Use Separate SSL Server Certificates” on page 224.

If you configure the Certificate Manager for SSL-enabled LDAP publishing, it also uses its SSL server certificate for SSL client authentication to the publishing directory; this is the default configuration. You can configure the Certificate Manager to use an alternate certificate for this purpose; see “Getting an SSL Client Certificate for a Subsystem” on page 226.

If you configure the Certificate Manager to function as a *trusted manager* to a Data Recovery Manager, the Certificate Manager also uses its SSL server certificate for SSL client authentication to the Data Recovery Manager. For details on trusted managers, see “Trusted Managers” on page 141. You can also configure the Certificate Manager to use an alternate certificate for this purpose; see “Getting an SSL Client Certificate for a Subsystem” on page 226.

Like any certificate, the SSL server certificate has a validity period. You must renew the certificate before it expires. For instructions on renewing the certificate, see “Renewing Certificates for the Subsystems” on page 239.

The SSL server key pair must be well protected to ensure that it is never compromised. However, if you know or suspect that the key pair has been compromised, reissue the certificate with a new key pair. For instructions on getting a new certificate, see “Getting New Certificates for the Subsystems” on page 232.

**Note** If you have installed the Certificate Manager with a Data Recovery Manager, both subsystems use the same SSL server certificate.

## Registration Manager’s Key Pairs and Certificates

The Registration Manager uses the following certificates:

- Signing Key Pair and Certificate
- SSL Server Key Pair and Certificate

## Signing Key Pair and Certificate

Every Registration Manager you have installed has a certificate, identified as the *Registration Manager signing certificate*, whose public key corresponds to the private key the Registration Manager uses to sign certificate requests before sending them to the Certificate Manager for signing. The Registration Manager's signature provides persistent proof to the Certificate Manager that the Registration Manager has processed the request. The first time you generated this certificate is when you installed the Registration Manager. The default nickname for the certificate is `raSigningCert cert-<instance_id>`, where `<instance_id>` identifies the CMS instance in which the Registration Manager is installed.

The Registration Manager's signing certificate was issued by the CA to which you submitted the certificate signing request. You might have submitted the request to an internally deployed CA or a public CA. To find out the issuer name, follow the instructions in "Viewing the Certificate Database Contents" on page 247.

If you configure the Registration Manager to function as a *trusted manager* to another subsystem, the Registration Manager uses its signing certificate for SSL client authentication to the subsystem; this is the default configuration. For details, see "Trusted Manager's Certificate for SSL Client Authentication" on page 144.

Like any certificate, the signing certificate has a validity period. You must renew the certificate before it expires. For instructions on renewing the certificate, see "Renewing Certificates for the Subsystems" on page 239.

The Registration Manager's signing key pair must be well protected to ensure that it is never compromised. However, if you know or suspect that the key pair has been compromised, reissue the certificate with a new key pair. For instructions on getting a new certificate, see "Getting New Certificates for the Subsystems" on page 232.

## SSL Server Key Pair and Certificate

Every Registration Manager you have installed has at least one *SSL server certificate*. The first time you generated this certificate is when you installed the Registration Manager. The default nickname for the certificate is `Server-Cert cert-<instance_id>`, where `<instance_id>` identifies the CMS instance in which the Registration Manager is installed.

The Registration Manager's SSL server certificate was issued by the CA to which you submitted the certificate signing request. You might have submitted the request to an internally deployed CA or a public CA. To find out the issuer name, follow the instructions in "Viewing the Certificate Database Contents" on page 247.

The Registration Manager uses its SSL server certificate to do SSL server-side authentication to the following:

- Netscape Console
- The end entity services interface (the HTTPS port)
- The Registration Manager Agent Services interface

By default, the Registration Manager uses a single SSL server certificate for authentication purposes. However, you can request and install additional SSL server certificates for the Registration Manager. For example, you can configure the Registration Manager to use separate server certificates for authenticating to Netscape Console, the end entity services interface, and the Registration Manager Agent Services interface. For instructions, see "Configuring the Server to Use Separate SSL Server Certificates" on page 224.

If you configure the Registration Manager for SSL-enabled LDAP publishing, it uses its SSL server certificate for SSL client authentication to the publishing directory; this is the default configuration. You can configure the Registration Manager to use an alternate certificate for this purpose. See "Getting an SSL Client Certificate for a Subsystem" on page 226.

Like any certificate, the SSL server certificate has a validity period. You must renew the certificate before it expires. For instructions on renewing the certificate, see "Renewing Certificates for the Subsystems" on page 239.

The SSL server key pair must be well protected to ensure that it is never compromised. However, if you know or suspect that the key pair has been compromised, reissue the certificate with a new key pair. For instructions on getting a new certificate, see "Getting New Certificates for the Subsystems" on page 232.

**Note** If you installed the Registration Manager with a Data Recovery Manager, both subsystems use the same SSL server certificate.

## Data Recovery Manager's Key Pairs and Certificates

The Data Recovery Manager uses the following certificates:

- Transport Key Pair and Certificate
- Storage Key Pair
- SSL Server Key Pair and Certificate

### Transport Key Pair and Certificate

Every Data Recovery Manager you have installed has a *Data Recovery Manager transport certificate*. The public key of the key pair that is used to generate the transport certificate is used by the client software to encrypt an end user's *encryption private key* before it is sent to the Data Recovery Manager for archival; only those clients capable of generating dual-key pairs (one for signing and one for encryption) use the transport certificate. For more information on how this certificate is used, see “Key Archival Process” on page 626.

The first time you generated this certificate is when you installed the Data Recovery Manager. The default nickname for the certificate is `kraTransportCert cert-<instance_id>`, where `<instance_id>` identifies the CMS instance in which the Data Recovery Manager is installed.

The transport certificate was issued by the CA to which you submitted the certificate signing request. You might have submitted the request to the Certificate Manager that is installed in the same instance, internally deployed another CA, or a public CA. To find out the issuer name, follow the instructions in “Viewing the Certificate Database Contents” on page 247.

Like any certificate, the transport certificate has a validity period. You must renew the certificate before it expires. For instructions on renewing the certificate, see “Renewing Certificates for the Subsystems” on page 239.

The transport key pair must be well protected to ensure that it is never compromised. However, if you know or suspect that the key pair has been compromised, reissue the certificate with a new key pair. For instructions on getting a new certificate, see “Getting New Certificates for the Subsystems” on page 232.

## Storage Key Pair

Every Data Recovery Manager you have installed has a *Data Recovery Manager storage key pair*. The first time you generated this key pair is when you installed the Data Recovery Manager.

The Data Recovery Manager uses the public component of this key pair to encrypt (or wrap) end users' encryption private keys during the key archival operation; it uses the private component to decrypt (or unwrap) the archived key during the recovery operation. That is, the private key encrypts the key repository it uses to store users' encryption private keys. For more information on how this key pair is used, see "Recovering Encrypted Data" on page 623.

The public component of the storage key pair is not certified; there is no certificate that corresponds to the public key.

Keys encrypted with the storage key can be retrieved only by authorized key recovery agents. For details, see "Key Recovery Agents and Their Passwords" on page 630.

## SSL Server Key Pair and Certificate

Every Data Recovery Manager you have installed has at least one *SSL server certificate*. The first time you generated this certificate is when you installed the Data Recovery Manager. The default nickname for the certificate is `Server-Cert cert-<instance_id>`, where `<instance_id>` identifies the CMS instance in which the Data Recovery Manager is installed.

The Data Recovery Manager's SSL server certificate was issued by the CA to which you submitted the certificate signing request. You might have submitted the request to the Certificate Manager that is installed in the same instance, an internally deployed CA, or a public CA. To find out the issuer name, follow the instructions in "Viewing the Certificate Database Contents" on page 247.

The Data Recovery Manager uses its SSL server certificate to do SSL server-side authentication to the following:

- Netscape Console
- The end entity services interface (the HTTPS port)
- The Data Recovery Manager Agent Services interface

By default, the Data Recovery Manager uses a single SSL server certificate for authentication purposes. However, you can request and install additional SSL server certificates for the Data Recovery Manager. For example, you can configure the Data Recovery Manager to use separate server certificates for authenticating to Netscape Console, the end entity services interface, and the Data Recovery Manager Agent Services interface. For instructions, see “Configuring the Server to Use Separate SSL Server Certificates” on page 224.

Like any certificate, the SSL server certificate has a validity period. You must renew the certificate before it expires. For instructions on renewing the certificate, see “Renewing Certificates for the Subsystems” on page 239.

If you know or suspect that the key pair for the SSL server certificate has been compromised, reissue the certificate with a new key pair. For instructions on getting a new certificate, see “Getting New Certificates for the Subsystems” on page 232.

**Note** If you installed the Data Recovery Manager with a Certificate Manager or Registration Manager, both subsystems use the same SSL server certificate.

## Tokens for Storing Keys and Certificates

A token is a hardware or software device that performs cryptographic functions and optionally stores public-key certificates, cryptographic keys, and data defined by the application using the cryptographic services. Alternatively, a token can also be considered as a device that you can use to generate and store your certificates, public and private key pairs, and associated information.

Certificate Management System defines two types of tokens, *internal* and *external*, for storing key pairs and certificates that belong to the Certificate Manager, Registration Manager, and Data Recovery Manager.

**Note** Only those who have the password that protects a token can access it. For information on changing the password that protects a token, use the command-line utility called the *Key Database Tool*; for details about this utility, see Appendix E, “Key Database Tool” in the online version of this document. To locate the document, see “Where to Go for Related Information” on page 27.



## Internal Token

An internal (software) token refers to a pair of software files, usually called *certificate database* and *key database*, that Certificate Management System uses to generate and store its key pairs and certificates. Certificate Management System automatically generates these files in the file system of its host machine when you choose to use the internal token for the first time. These files were created for you during CMS installation if you chose to use the internal token for key-pair generation.

In the CMS host system, the certificate database is identified by the name `cert7.db`; the key database is identified by the name `key3.db`. You can find both these files at this location:

```
<server_root>/cert-<instance_id>/config/...
```

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

`<instance_id>` is the ID for this instance of Certificate Management System. You first specified this when you installed this server.

## External Token

An external (hardware) token refers to an external hardware device, such as a smart card, FORTEZZA card, or other crypto card, that Certificate Management System uses to generate and store its key pairs and certificates. Certificate Management System supports any hardware tokens that are compliant with PKCS #11 version 2.01. For details, see the information provided at this URL:

```
http://developer.netscape.com/support/faqs/pkcs\_11.html
```

If you haven't already done so, consider using external tokens for generating and storing the key pairs and certificates used by Certificate Management System. These devices represent another security measure you can take to safeguard private keys.

## Installing External Tokens

To use external encryption devices or tokens, you need to take the following steps:

- Step 1. Install the Cryptographic Device
- Step 2. Install the PKCS #11 Module

### Step 1. Install the Cryptographic Device

To install the drivers provided by the device manufacturer, follow the instructions that came with the device. When you install a hardware token, you are given an opportunity to name it; be sure to use a name that will help you identify the token later.

### Step 2. Install the PKCS #11 Module

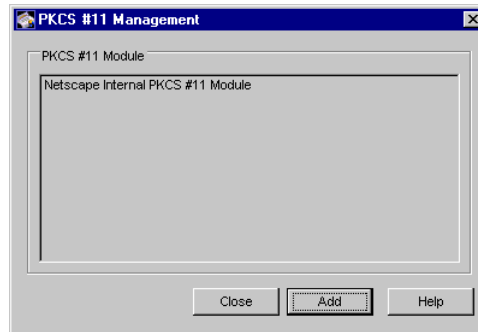
PKCS #11 is a standard set of APIs and shared libraries used by Netscape and a number of encryption vendors. PKCS #11 isolates an application from the details of the cryptographic device, thus enabling the application to provide a unified interface for PKCS #11-compliant cryptographic devices.

The PKCS #11 module implemented in Certificate Management System (in Netscape Administration Server) enables it to support cryptographic devices supplied by many different manufacturers. Specifically, it allows Certificate Management System to plug in shared libraries or DLLs supplied by manufacturers of external encryption devices and use them for generating and storing keys and certificates for the Certificate Manager, Registration Manager, and Data Recovery Manager.

You install the PKCS #11 module in one of the two ways, depending on how you received the DLLs for the cryptographic device.

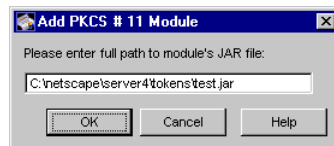
- If the device vendor provided the DLLs as a JAR file, use Netscape Console to install the PKCS #11 module:
  1. Access the CMS window (see “Accessing the CMS Window” on page 63).
  2. From the Console menu, choose Manage PKCS#11.

The PKCS #11 Management window appears.



3. Click Add.

The Add PKCS #11 Module window appears.



4. Enter the path to the JAR file you obtained in Step 1.
5. Click OK.

- If the device vendor provided the DLLs (not as a JAR file), use the command-line tool called `modutil` to install the PKCS #11 module. This tool is located here:

```
<server_root>/shared/bin/modutil
```

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

1. Locate the CMS instance for which you want to install the PKCS #11 module.
2. Open a terminal window.
3. Go to the configuration directory of Administration Server; it is located here:

```
<server_root>/admin-serv/config
```

4. At the prompt, enter this command:

```
<server-root>/shared/bin/modutil -dbdir . -nocertdb
-create
```

This creates the required security module database file (`secmod.db`) in the configuration directory.

5. At the prompt, enter this command:

```
<server_root>/shared/bin/modutil -dbdir . -nocertdb
-add <module_name> -libfile <library_file>
```

`<library_file>` specifies the path to the DLL or other library file containing the implementation of the PKCS #11 interface module.

`<module_name>` specifies the name of the PKCS #11 module (which you specified in Step 1 when you installed the drivers).

For example, if you are installing a Litronic token, the command would look like this:

```
<server_root>/shared/bin/modutil -dbdir . -nocertdb
-add CryptOS -libfile core32
```

6. Copy the new `secmod.db` (`secmodule.db` on Unix) to the following location:

```
<server-root>/cert-<instance_id>/config
```

This overwrites the old `secmod.db` in this directory.

## Managing Tokens Used by the Subsystems

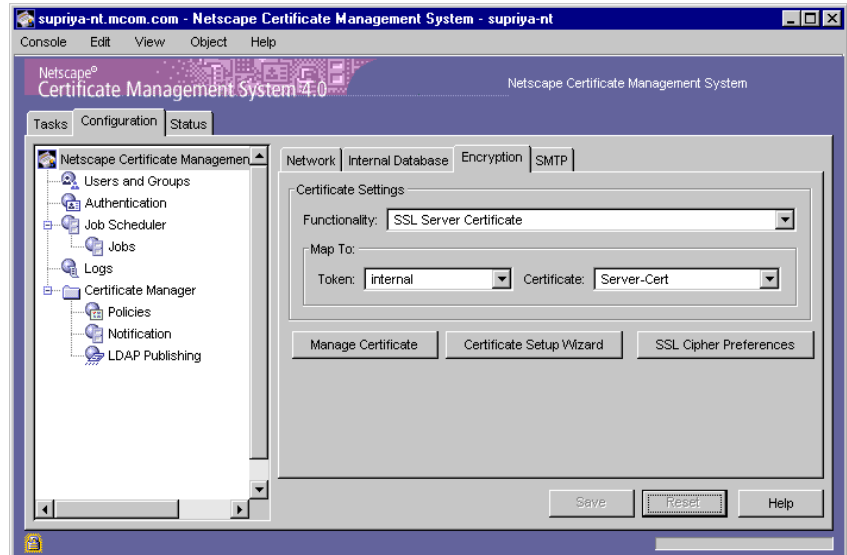
There are two main tasks involved in managing the tokens used by Certificate Management System:

- Viewing Tokens
- Changing a Token's Password

### Viewing Tokens

To view a list of the tokens currently installed for a CMS instance:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab, and then in the right pane, click the Encryption tab.



3. In the Map To section, check the Token drop-down list.

It shows the names (as specified when the tokens were installed) of external tokens installed for the currently selected CMS instance. For information on installing external tokens, see “External Token” on page 193.

## Changing a Token’s Password

The token, internal or external, that stores the key pairs and certificates for the subsystems is protected (encrypted) by a password. To decrypt the key pairs or to gain access to them, you must enter that password. The first time you specified this password is when you used the token the first time, most likely during CMS installation.

It is good security practice to periodically change the password that protects your server’s keys and certificates; changing the password periodically minimizes the risk of someone finding out the password. To change a token’s password, use the command-line utility called the *Key Database Tool*; for details about this utility, see Appendix E, “Key Database Tool” in the online version of this document.

Note that the single sign-on password cache stores the passwords for tokens in order to start the server using a single password; for details, see “Required Start-up Information” on page 106. Whenever you change the password, the cache is updated with the new password.

# Hardware Cryptographic Accelerators

Certificate Management System allows you to use hardware cryptographic accelerators with external tokens. Many of the accelerators provide the following security features:

- Fast SSL connections. Speed is important if you want your Certificate Manager, Registration Manager, or Data Recovery Manager to be able to accommodate a high number of simultaneous enrollment or service requests.

- Hardware protection of private keys. These devices behave like smart cards, in that they do not allow the private keys to be removed from the hardware. This is important if you are concerned about the risks associated with key theft from an active attacker of your online Registration Manager or Certificate Manager.

For a list of vendors that supply hardware cryptographic accelerators, see this URL:

<http://www.netscape.com/cms/v4.0/index.html>

## Certificate Setup Wizard

Certificate Management System provides a wizard, called the *Certificate Setup Wizard*, that automates the process of requesting and installing the certificates required by Certificate Management System. (For a description of these certificates, see “Keys and Certificates for the Main Subsystems” on page 184.)

The Certificate Setup Wizard is integrated into the CMS window and allows you to accomplish the following tasks:

- Request and install certificates (based on new or existing key pairs stored in internal and external tokens) for the Certificate Manager, Registration Manager, and Data Recovery Manager installed in the currently selected CMS instance
- Install CA certificates in the trust database of the currently selected CMS instance
- Install CA certificate chains in the trust database of the currently selected CMS instance

When you start the wizard, which you do by clicking the Certificate Setup Wizard button in the Encryption tab of the CMS window (see the figure on page 197), you are asked to specify whether you want to request or install a certificate. The wizard presents you with the screens appropriate to your choice and walks you through the entire process.

For installing certificates, except for cases when the certificate is self-signed by the CA, you will need to run the wizard twice: once, to request the certificate and once to install the certificate. The reason for this is, if you submit the certificate request to a non-local CA, you will have to wait for the certificate until it is delivered to you.

The following sections explain the process of requesting and installing a certificate by using the Certificate Setup Wizard:

- Using the Wizard to Request a Certificate
- Using the Wizard to Install a Certificate or Certificate Chain

For instructions on getting new certificates, see “Getting New Certificates for the Subsystems” on page 232. For instructions on renewing existing certificates, see “Renewing Certificates for the Subsystems” on page 239.

## Using the Wizard to Request a Certificate

The Certificate Setup Wizard allows you to request any of the certificates used by the Certificate Manager, Registration Manager, and Data Recovery Manager installed in the currently selected CMS instance.

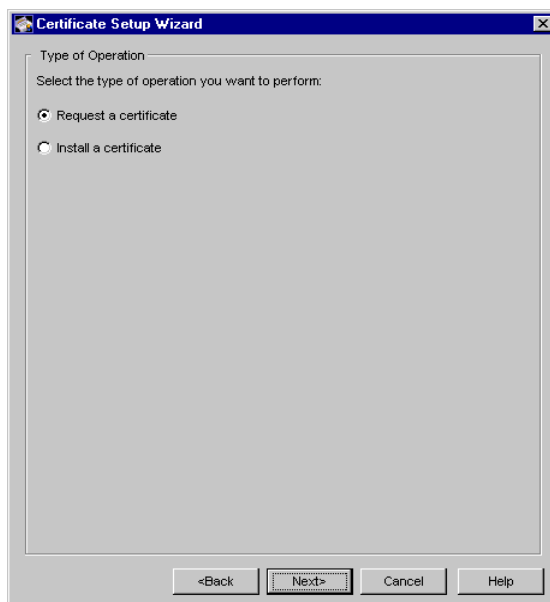
Using the wizard to request a certificate involves the following steps:

- Step 1. Select the Operation
- Step 2. Choose the Certificate
- Step 3. Specify the Key-Pair Information
- Step 4. Specify the Subject Name for the Certificate
- Step 5. Specify the Validity Period
- Step 6. Specify Extensions
- Step 7. Copy the Certificate Signing Request
- Step 9. Send the Certificate Signing Request to a CA
- Step 8. Check the Certificate Request Status



## Step I. Select the Operation

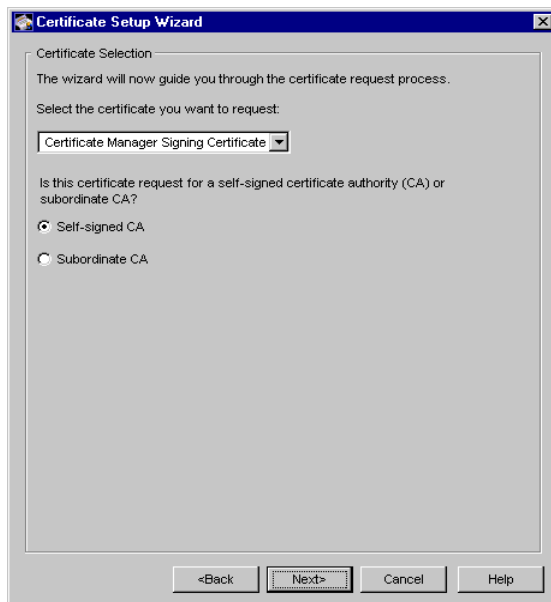
Indicate whether you want to request a certificate or install a certificate.



For the sake of completing the instructions that follow, assume that you chose to request a certificate.

## Step 2. Choose the Certificate

Choose the certificate (by name) that you want to request.



The drop-down list shows various certificates used by the currently selected CMS instance. Choose the one you want to request. (If you need information on certificates used by Certificate Management System, see “Keys and Certificates for the Main Subsystems” on page 184.)

Which certificates you see in the list depends on the subsystems installed in the currently selected CMS instance. You may see a combination of the following options:

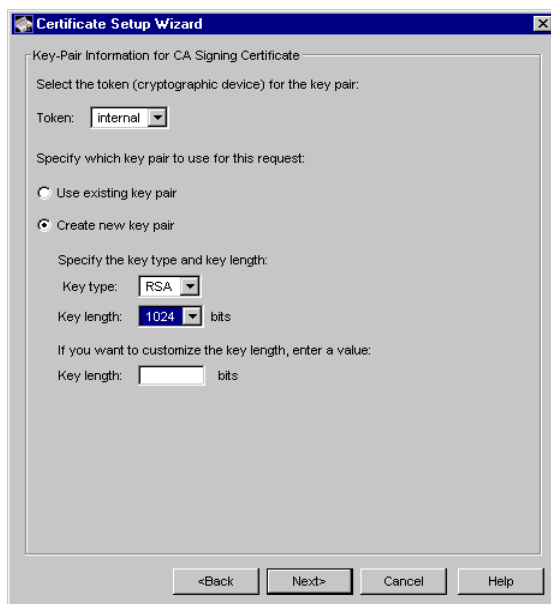
- If a Certificate Manager is installed, the list includes the Certificate Manager’s CA signing and SSL server certificates.
- If a Registration Manager is installed, the list includes the Registration Manager’s signing and SSL server certificates.
- If a Data Recovery Manager is installed, the list includes the Data Recovery Manager’s transport and SSL server certificate.

Depending on the certificate you want to generate, choose the one in the drop-down list:

- **Certificate Manager Signing Certificate**—choose this option if you want to request a signing certificate for the Certificate Manager installed in the currently selected CMS instance. If you choose this option, you must also specify whether the certificate request is for a self-signed CA (also known as the root CA) or a subordinate CA.
- **Registration Manager Signing Certificate**—choose this option if you want to request a signing certificate for the Registration Manager installed in the currently selected CMS instance.
- **Data Recovery Manager Transport Certificate**—choose this option if you want to request a transport certificate for the Data Recovery Manager installed in the currently selected CMS instance.
- **SSL Server Certificate**—choose this option if you want to generate an SSL server certificate request for the Certificate Manager, Registration Manager, or Data Recovery Manager installed in the currently selected CMS instance.

### Step 3. Specify the Key-Pair Information

Specify the key-pair information for the certificate to be requested.



You need to identify the following:

- The token that contains the key pair for generating the certificate request

The drop-down list shows the names of tokens currently installed for the selected CMS instance; these are the tokens you can use now.

- The internal token is identified as *internal*. You should choose this option if the key pair for the certificate you chose in the previous step is stored in the local key database.
- The names of external tokens vary, matching the names specified when the tokens were installed. You should choose this option if the key pair for the certificate you chose in the previous step is in an external cryptographic device. If you don't see the token you want to use, exit from the wizard, make sure the token is installed properly, restart the server, and repeat the process. For information on using or installing external tokens, see "Installing External Tokens" on page 194.

- The key pair for generating the certificate request

You can choose to generate the certificate request based on an existing or a new key pair.

- If you want to renew the certificate you selected in the previous step, use the existing key pair for generating the request. For example, you can extend the validity period of a certificate by renewing it.

To generate a certificate request based on an existing key pair, select the token that contains the key pair you want to use for generating the request. The wizard automatically selects the key pair that corresponds to the certificate you chose in the previous step.

- If you want a new certificate, use a new key pair for generating the request. For example, you may want to get a new SSL server certificate or may want to replace an existing certificate whose private key has been compromised.

To generate a certificate request based on a new key pair, select the token that can generate the key pair you want to use for generating the request. For example, if you want to generate the key pair using an external cryptographic device, such as a smart card, select that as the token. In addition, you will be required to indicate details, such as the key algorithm and size for the key pair.

- The type and length of the key pair

You are required to provide this information only if you chose to generate the certificate request based on a new key pair. For key type, you can choose RSA or DSA. Be sure to select a key type that the CA (to which you will later submit the request for signing) can certify.

For key length, enter the size in bits. The choices for RSA key type are 512, 1024, 2048, or custom. The choices for DSA key type are 512, 1024, or custom. Keep in mind that generating a new key pair takes time—the longer the key length the longer the time the wizard takes to generate it.

## Step 4. Specify the Subject Name for the Certificate

Specify the subject name, in distinguished name (DN) format, for the certificate to be requested.

**Certificate Setup Wizard**

Subject Name for CA Signing Certificate

The current subject name in distinguished name (DN) format:  
CN=MyTestCA, OU=MC, O=Netscape, L=MV, ST=CA, C=US

To modify the subject DN for the certificate.

☒ Enter the values for the subject DN components:

Common Name (CN=):

Organizational Unit (OU=):

Organization (O=):

Locality (L=):

State (ST=):

Country (C=):

Selected DN: CN=MyTestCA, OU=Marketing, O=Netscape Comm Corp, L=Mountain View, ST=CA, C=US

☐ Enter the values for the subject DN string:

<Back   Next>   Cancel   Help

You can either enter values for individual DN attributes required to build the subject DN or build the complete subject DN string yourself. If you enter values for individual DN attributes, the wizard constructs the subject DN string.

If you want to enter values for individual DN components, provide the following information:

- Common name—enter the name as appropriate. The common name format, except for the SSL server certificate, can be a descriptive name of up to 255 characters; for example, you can name the Certificate Manager's signing certificate as "Root CA for ABC Corporation"; similarly, you can name the Registration Manager's signing certificate as "Registration Authority for North America". For a SSL server certificate, the name must be the fully qualified host name of Certificate Management System in this form:

```
<machine_name>.<your_domain>.<domain>
```

To determine the machine and domain names, go to Netscape Console, and locate the CMS host in the navigation tree.

- Organizational unit—enter the organizational unit you belong to.
- Organization—enter a description that identifies your organization.
- Locality—enter the name of the city where your business is located.
- State or province—enter the name of the state or province where your business is located.
- Country—enter the name of the country where your business is located.

**Important** When choosing subject names for certificates, be sure to follow the information provided in “Role of Distinguished Names in Certificates” on page 662.

## Step 5. Specify the Validity Period

You need to complete this step only if you chose to generate a self-signed CA certificate request.

**Certificate Setup Wizard**

Validity Period for CA Signing Certificate

Specify the validity period for the certificate:

	YYYY	MM	DD	HH	mm	SS
Begin on:	1999	01	05	10	30	00
Expire on:	2002	01	05	10	30	00

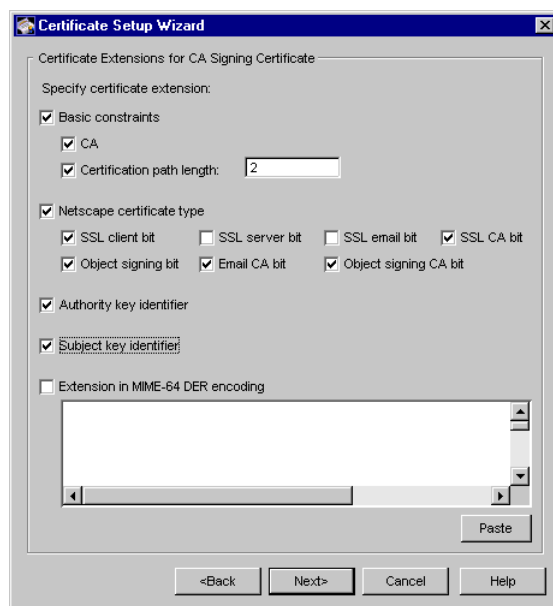
<Back   Next>   Cancel   Help

Specify the starting and ending dates of the validity period for the certificate request you want to generate. You can also specify the time at which the validity period should start and end on those dates. The date and time are in the form YYYYMMDD (Year, Month, Day) and HHmmSS (Hour, Minute, Second), in that order.

The default validity period is one year.

## Step 6. Specify Extensions

You need to complete this step only if you chose to generate a CA signing certificate request for a Certificate Manager (deployed as either the root CA or a subordinate CA).



This screen allows you to set the standard X.509 version 3 extensions and Netscape-defined extensions for the certificate to be requested. The required extensions are chosen by default. If you want to change the default choices, be sure to read the general guidelines explained in "Certificate Extensions" in Appendix B of *Netscape Certificate Management System Installation and Deployment Guide*.



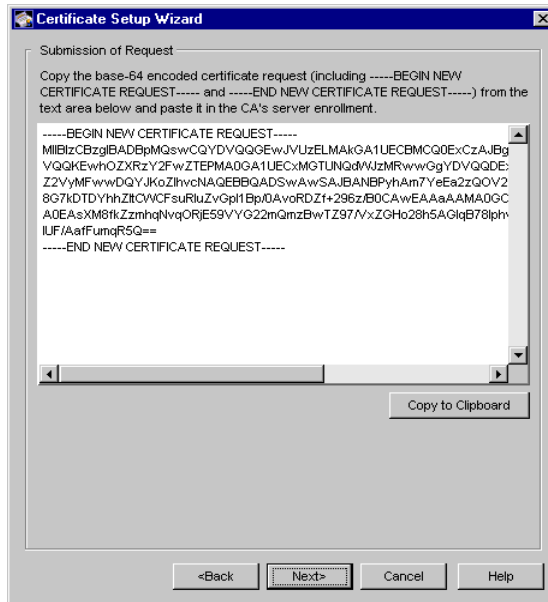
Also note that certificate extensions are required if you are setting up a hierarchy of certificate authorities (CAs). Subordinate CAs must have certificates that include the extension identifying them as either a subordinate SSL CA (which allows them to issue certificates for SSL) or a subordinate email CA (which allows them to issue certificates for secure email). If you disable certificate extensions, you will not be able to set up CA hierarchies. For more information on CA hierarchies, see "Certificate Hierarchies" in Appendix D of *Managing Servers with Netscape Console*.

You can set the following extensions:

- Basic constraints—select this option if you want to set any of the Basic Constraints extension bits in the certificate you are requesting. When you select the option, the associated fields are enabled. You should select the ones you want to set.
- Netscape certificate type—select this option if you want to set any of the Netscape Certificate Type extension bits in the certificate you are requesting. When you select the option, the associated fields are enabled. You should select the ones you want to set.
- Authority key identifier—select this option if you want to set the authority key identifier extension.
- Subject key identifier—select this option if you want to set the subject key identifier extension.
- Key usage—select this option if you want to set the key usage extension. If you choose this option, the digital signature (bit 0), non repudiation (bit 1), key Certificate Sign (bit 5), and CRL sign (bit 6) bits are set by default. The extension is marked critical as recommended by the PKIX standard and rfc 2459 (see <ftp://ftp.isi.edu/in-notes/rfc2459.txt> for a description of the Key Usage extension).
- Extension in MIME 64 DER encoding—select this option if you want to specify any custom extension. When you select the option, the associated text field is enabled. You should paste your extension (in MIME 64 DER encoded format) into the text field.

## Step 7. Copy the Certificate Signing Request

Based on the information you've entered in the previous steps, the wizard now displays the certificate signing request (CSR).



The request is in a base-64 encoded PKCS #10 format and is bounded by the marker lines -----BEGIN NEW CERTIFICATE REQUEST----- and -----END NEW CERTIFICATE REQUEST-----.

An example is show below:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
```

```
MIICJzCCAZCgAwIBAgIBAzANBgkqhkiG9w0BAQQFADBC6SAwHgYDVQQKEXdOZXRzY2FwZSBDb21td
W5pY2F0aW9uc2ngjhnMVQ2VydG1maWNhdGUgQXV0aG9yaXR5MB4XDtk4MDgyNzE5MDAwMFOxDTk5M
DIyMzE5MDAwMnbjdngYoxIDAeBgNVBAoTF05ldHNjYXB1IENvbnV1bW11bmljYXRpb25zMQ8wDQYDVQQ
LEWZQZW9wbGUxZjZAVBgoJkiaJk1sZAEwEwddzdXByaX1hMRcwFQYDVQQDEW5TdXByaX1hIFNoZXR0e
TEjMCEGCSqGS1b3DbndgJARVUc3Vwcm15Yhvfggsvwryw4y7214vAOBgNVHQ8BAf8EBAMCBLAwFAY
JYIZIAyb4QgEBAQHBAQDAGCAMA0GCSqGS1b3DQEBAUAA4GBAFi9FzyJlLmS+kzsue0kTXawbwamG
dYq12w4hIBgdR+jWeLmD4CP4xzmKdvQ6IqD2q8DBs91RQu9JYg129oaCLpZfMNTpMnc3WPKO2pWZW
Um7waHEtdbo9vSpbJkXTM/2GhWbs05vLdeOxrPGxiHkHgV/
vmqCl4HW7AorqGgyfygbhgtgutrhyj
```

```
-----END NEW CERTIFICATE REQUEST-----
```

Do not modify the CSR; you must send it to the CA as is.

Copy the CSR, including the marker lines -----BEGIN NEW CERTIFICATE REQUEST----- and -----END NEW CERTIFICATE REQUEST-----, to a text file. If you are running the wizard on a Windows NT system, you can also copy the CSR to the Windows clipboard. In a Unix system, you may have to open an application, such as Netscape Composer, with a clipboard.

The wizard also copies the CSR to a text file it creates in the configuration directory, which is located at

```
<server_root>/cert-<instance_id>/config/...
```

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

`<instance_id>` is the ID for this instance of Certificate Management System. You first specified this when you installed this server.

The name of the text file varies depending on for which key pair you generated the CSR:

- If the CSR is for a Certificate Manager CA signing certificate, the file name will be `cacsr.txt`.
- If the CSR is for a Registration Manager signing certificate, the file name will be `racsr.txt`.
- If the CSR is for a Data Recovery Manager transport certificate, the file name will be `kracsr.txt`.
- If the CSR is for a SSL server certificate, the file name will be `sslcsr.txt`.

## Step 8. Check the Certificate Request Status

The wizard now informs you of the status of the request.

- If you requested a self-signed CA certificate, the wizard automatically submits the CSR to the CA. If the CSR includes all the required information, the CA signs the certificate and returns it to the wizard, which then installs it in the appropriate token.
- If you requested any other certificate, you must submit the CSR to a CA of your choice manually. This is explained in Step 9.

## Step 9. Send the Certificate Signing Request to a CA

This step is not part of the wizard process. Follow these instructions to manually send the CSR to a CA of your choice. The CA can be internal or external.



- An internal CA is any CA deployed internally by your organization—for example, a Certificate Manager installed in the same or another instance of Certificate Management System. See “Sending the CSR to an Internal CA” on page 212.
- An external CA is any public CA. Before sending the CSR to a public CA, make sure that the CA can issue the certificate you want to request. Also, it is a good idea to read the policy statement published by a CA to see whether the CA imposes any restrictions on the validity period or usage of the certificate. See “Sending the CSR to an External CA” on page 214.

### Sending the CSR to an Internal CA

The following instructions assume that your internally deployed CA is a Certificate Manager and that you are using the default HTML forms provided for end-entity enrollment. If you have customized these forms, you should follow the appropriate instructions.

To send the CSR manually to an internal CA:

1. Locate the text file to which you copied the CSR earlier (see “Step 7. Copy the Certificate Signing Request” on page 210).
2. Open a web browser window.
3. Enter the URL to the CA’s home page.

By default, the CA’s home page is the end entity services interface. Depending on the port at which the CA is listening to end-entity requests (see “End-Entity Ports” on page 124) the URL to the end entity services is one of the following:

`http://<host_name>:<end_entity_port>`

or

`https://<host_name>:<end_entity_port>`

where <host\_name> is in the form  
`<machine_name>.<your_domain>.<domain>`

The end entity services interface appears.

4. Click the Enrollment tab.
5. In the menu list, click the appropriate link:
  - If the CSR is for a subordinate CA certificate, select the manual enrollment link in the Certificate Manager Enrollment section.
  - If the CSR is for a Registration Manager’s signing certificate, select the manual enrollment link in the Registration Manager Enrollment section.
  - If the CSR is for an SSL server certificate, select either the manual or the directory-based link in the Server Enrollment section.

For information on how the default server enrollment forms work, see “Certificate Issuance to Servers” on page 603.

6. In the form that appears, enter the required information and paste the CSR from the text file.

Be sure to include the marker lines, -----BEGIN NEW CERTIFICATE REQUEST----- and -----END NEW CERTIFICATE REQUEST-----.

7. Submit the request.
8. When the CA sends you a response, save the information in a text file for future reference or inquiry.
  - If you submitted the request using a manual enrollment form, the request gets queued.

An agent needs to process and approve the certificate request, which the CA signs then and delivers back to the email address specified in the request. You can contact the CA agent to find out when the certificate will be delivered to you.
  - If you submitted the request using an automated enrollment form, the request gets processed automatically by the CA.

Keep in mind that the CA can reject or queue the request if it fails any of the authentication or policy checks. A request that gets queued must be processed manually by an agent.
9. When you receive the certificate from the CA, install it following the instructions in “Using the Wizard to Install a Certificate or Certificate Chain” on page 215.

## **Sending the CSR to an External CA**

To send the CSR manually to an external CA:

1. Locate the text file to which you copied the CSR (see “Step 7. Copy the Certificate Signing Request” on page 210).
2. Open a web browser window.
3. Navigate to the CA’s home page by entering the appropriate URL in the browser window.
4. Locate the form that allows you to submit certificate requests for servers.

5. Enter the required information and paste the CSR from the text file.

Be sure to include the marker lines, -----BEGIN NEW CERTIFICATE REQUEST----- and -----END NEW CERTIFICATE REQUEST-----.

6. Submit the request.
7. When the CA sends you a response, save the information in a text file for future reference or inquiry.
8. When you receive the certificate from the CA, install it following the instructions in “Using the Wizard to Install a Certificate or Certificate Chain” on page 215.

## Using the Wizard to Install a Certificate or Certificate Chain

The Certificate Setup Wizard allows you to install or import the following certificates into either an internal or external token used by the currently selected CMS instance:

- Any of the certificates used by the Certificate Manager, Registration Manager, and Data Recovery Manager installed in the currently selected CMS instance
- Any other trusted CA certificates (certificates of CAs that you want to trust)
- Certificate chains

A certificate chain typically includes a collection of certificates: the subject certificate, the trusted root CA certificate, and any intermediate CA certificates needed to link the subject certificate to the trusted root. However, the certificate chain the wizard allows you to import must include only CA certificates; none of the certificates can be a user certificate.

In a certificate chain, each certificate in the chain is encoded as a separate DER-encoded object. When the wizard imports a certificate chain, it imports these objects one after the other, all the way up the chain to the last certificate, which may or may not be the root CA certificate. If any of the certificates in the chain already exist in the local certificate database, the

wizard replaces them by the ones included in the chain. If the chain includes intermediate CA certificates, the wizard adds them to the certificate database as *untrusted* CA certificates.

The certificate or certificate chain you provide to the wizard for installation must be in one of the data formats supported by the wizard. This is explained in “Data Formats for Installing Certificates and Certificate Chains” on page 216.

Using the wizard to install a certificate or certificate chain involves the following steps, described in detail on page 218:

- Step 1. Select the Operation
- Step 2. Select the Certificate or Certificate Chain
- Step 3. Specify the Location of the Certificate
- Step 4. View the Certificate or Certificate Chain
- Step 5. Install the Certificate or Certificate Chain
- Step 6. Verify the Certificate Status

## Data Formats for Installing Certificates and Certificate Chains

The wizard can accept certificates and certificate chains in several data formats. This section briefly explains the data formats recognized by the wizard.

### Binary Formats

The wizard can recognize certificates and certificate chains in the following binary formats:

- DER-encoded certificate

This is a single binary DER-encoded certificate.

- PKCS #7 SignedData objects

This is a PKCS #7 SignedData object. The only significant field in the SignedData object is the certificate. In particular, the signature and the contents are ignored. The PKCS #7 format allows multiple certificates to be downloaded at once.



- DER-encoded certificates

These are DER-encoded certificates that may or may not be wrapped in a base-64 encoding package surrounded by the delimiters -----BEGIN CERTIFICATE----- and -----END CERTIFICATE-----.

- Netscape Certificate Sequence

This is a simpler format for downloading certificate chains. It consists of a PKCS #7 ContentInfo structure, wrapping a sequence of certificates. The value of the contentType field should be netscape-cert-sequence, while the content field is the following structure:

```
CertificateSequence ::= SEQUENCE OF Certificate
```

This format allows multiple certificates to be downloaded at once.

## Text Formats

The wizard can also import certificates and certificate chains in text formats. Here's what you should be aware of when using the wizard to install a certificate or certificate chain in text format:

The text format must begin with the following line:

```
-----BEGIN CERTIFICATE-----
```

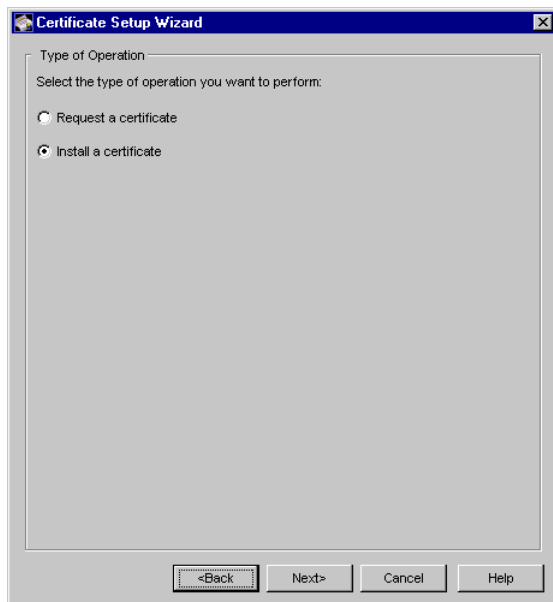
Following this line should be the certificate data, which can be in any of the binary formats described in “Binary Formats” on page 216. This data should be base-64 encoded as described by RFC 1113 (for details, see <http://www.scit.wlv.ac.uk/rfc/rfc11xx/RFC1113.html>).

Following the certificate data must be this line:

```
-----END CERTIFICATE-----
```

## Step I. Select the Operation

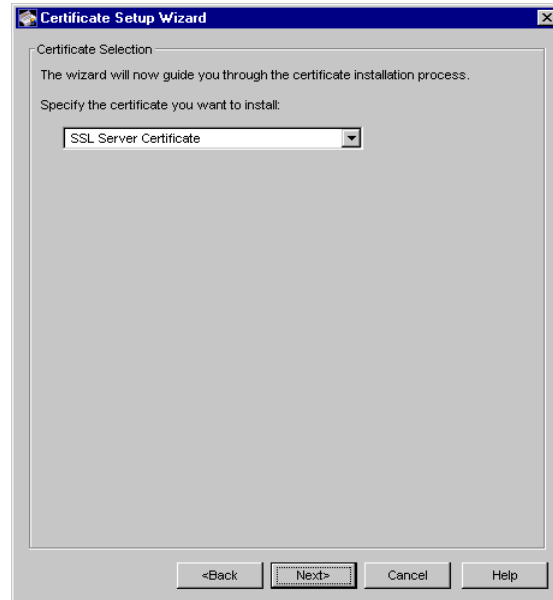
Indicate whether you want to request a certificate or install a certificate.



For the sake of completing the instructions that follow, assume that you chose to install a certificate.

## Step 2. Select the Certificate or Certificate Chain

Select the certificate you want to install.



The drop-down list shows various options:

- Any of the certificates used by the subsystems—Certificate Manager, Registration Manager, or Data Recovery Manager—installed in the currently selected CMS instance.
- Any other trusted CA certificates
- Certificate chains

Depending on whether you want to install a CMS certificate, any other trusted CA certificate, or a CA certificate chain, choose the appropriate option from the list box:

- Certificate Manager Signing Certificate—choose this option if you want to install a signing certificate for the Certificate Manager installed in the currently selected CMS instance.

- **Registration Manager Signing Certificate**—choose this option if you want to install a request signing certificate for the Registration Manager installed in the currently selected CMS instance.
- **Data Recovery Manager Transport Certificate**—choose this option if you want to install a transport certificate for the Data Recovery Manager installed in the currently selected CMS instance.
- **SSL Server Certificate**—choose this option if you want to install an SSL server certificate for the Certificate Manager, Registration Manager, or Data Recovery Manager installed in the currently selected CMS instance.
- **Server Certificate Chain**—choose this option if you want to install a CA certificate chain; the CA certificate will be included in the chain.
- **Other Trusted CA Certificate**—choose this option if you want to install a trusted CA certificate.

### Step 3. Specify the Location of the Certificate

Locate the certificate or certificate chain you want to install.

**Certificate Setup Wizard**

Location of Certificate

Indicate the location of the certificate:

☐ The certificate is located in this file:

☒ The certificate is located in the text area below:

Paste a base-64 encoded certificate (including -----BEGIN CERTIFICATE----- and -----END CERTIFICATE-----) in the text area.

```
-----BEGIN CERTIFICATE-----
MIB+TCCaAogAwIBAgIBATANBgkqhkiG9w0BAQoFADBpMQswCQYDVQQL
Q0ExCzAIBgNVBACtAk1VMREwDwYDVQQKEwhOZXRzY2FwZTEPMA0G
CgQExNDZlJ0aWZpY2F0ZSBhbnV5ShZ2YyMB4XDk5MDMyMjA4MDAwMl
aTELMAkGA1UEBhMCVVMxIzAIBgNVBAGTAkNBMQswCQYDVQQLQ0EwJN
cGUxDzANBgNVBAsTBk1DUHviczEcmBoGA1UEAxMTc3Vwcm5SYs1udCSt
Slt3DGEBAQUAA0sAMEgCQQCF3+Gt795w2FL4+0yyTs5+1yve300ASVQIK
-----END CERTIFICATE-----
```

Paste from Clipboard

<Back Next> Cancel Help

You can keep the certificate or certificate chain in a text file or copy it to the text area on the wizard screen. Here is some information that will help you decide on the location.

- Keeping the certificate or certificate chain in a text file

The wizard can import a certificate or certificate chain from a text file in *text* as well as *binary* formats; see “Data Formats for Installing Certificates and Certificate Chains” on page 216. If you have copied the certificate or certificate chain to a text file, you will be required to provide the wizard with the absolute path to that file. The file must be located in the host system the wizard is running. If the file is located elsewhere, exit from the wizard, copy the file to the local disk, and restart the wizard.

- Copying the certificate or certificate chain to the text area on the wizard screen

You can paste the certificate or certificate chain into the text area provided by the wizard. This is a text input field, so you can paste the certificate or certificate chain in text format only. For example, if you are installing a certificate, it base-64 encoded certificate blob should look similar to this:

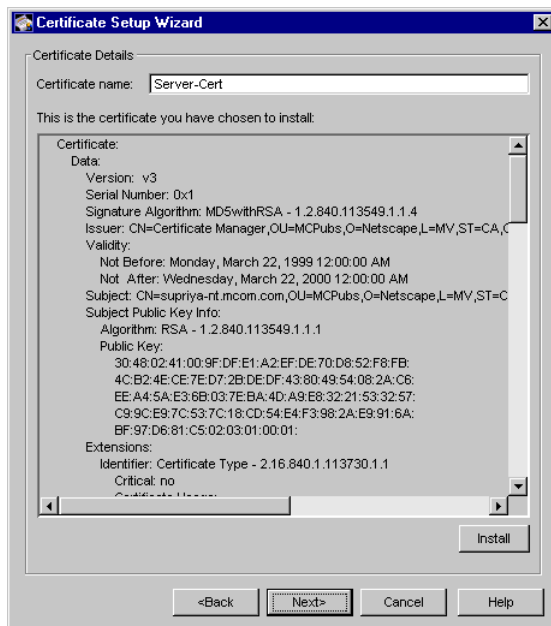
-----BEGIN CERTIFICATE-----

[illegible]

-----END CERTIFICATE-----

## Step 4. View the Certificate or Certificate Chain

The wizard displays the certificate or certificate chain you have chosen to install. Make sure you have chosen the right one; otherwise, use the Back button to go back and locate the right one. Specify a nickname for the certificate.



## Step 5. Install the Certificate or Certificate Chain

The wizard shows the certificate or certificate chain information you have selected for installing. You should check the information to make sure that you have chosen the correct one for installing.

After verifying that the certificate you have chosen is the correct one, click the Install button. The wizard installs the certificate or the CA chain in the token you have chosen.

- If you installed a certificate that has been issued by CA whose certificate chain doesn't exist in the certificate database, you must add that CA's certificate chain to the database. To add the CA chain to the database, copy the CA chain to a text file, start the wizard again, and install the CA chain.

- If you installed (or imported) a certificate chain, the wizard adds (to the local trust database) the first certificate in the chain as a trusted CA certificate and any subsequent certificates as untrusted CA certificates. For more information on how the wizard installs a certificate chain, see “Using the Wizard to Install a Certificate or Certificate Chain” on page 215.

## Step 6. Verify the Certificate Status

This step is applicable only if you installed a certificate chain.

After you install a certificate chain in the trust database of a CMS instance, check the trust status of each certificate that got installed, and make sure that the correct CA certificates are trusted. For instructions, see “Changing the Trust Settings of a CA Certificate” on page 250.

# Configuring the Server's Security Preferences

Configuring the server's security preferences involves identifying the following:

- The SSL server certificates the server must use for authenticating to the end entity, agent, and administration interfaces. For details, see “Configuring the Server to Use Separate SSL Server Certificates” on page 224.
- The SSL client certificate the server must use for authenticating to the publishing directory. For details, see “Getting an SSL Client Certificate for a Subsystem” on page 226.
- The version of SSL that an instance of Certificate Management System must use during SSL communication. The latest version is SSL version 3, but many older clients use SSL version 2. Because client authentication is required for performing privileged operations, you must enable SSL version 3 ciphers supported by Certificate Management System. For details, see “Setting Up Cipher Preferences for SSL Communications” on page 228.

## Configuring the Server to Use Separate SSL Server Certificates

You can configure a CMS instance to use separate SSL server certificates for authenticating to Netscape Console, the Agent Services interface, and the end entity services interface.

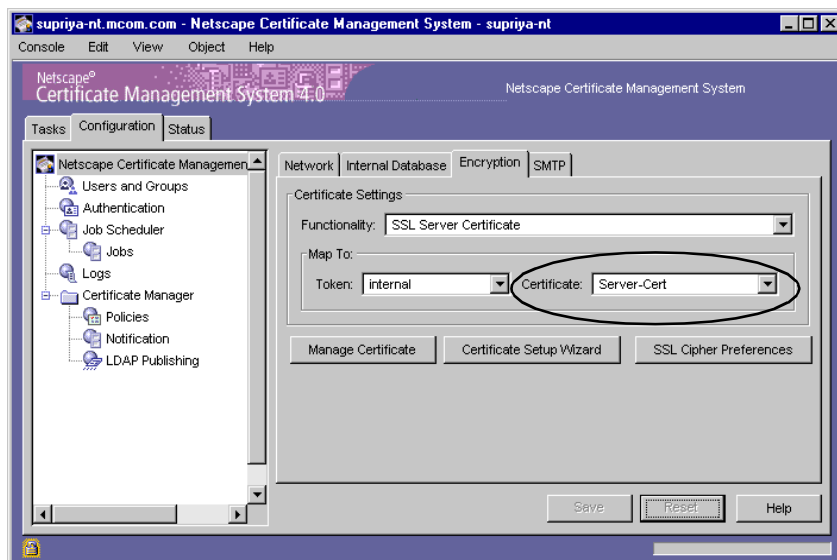
This configuration involves the following steps:

- Step 1. Get the Required SSL Server Certificates
- Step 2: Update the Configuration

### Step 1. Get the Required SSL Server Certificates

You must first request and install the required number of SSL server certificates for the particular CMS instance. For instructions, see “Getting New Certificates for the Subsystems” on page 232.

Once you have installed the certificates, you should be able to see them in the list of SSL server certificates in the Encryption tab of the CMS window.





## Step 2: Update the Configuration

After you verify that the certificates are installed, configure the server as follows:

1. Stop the CMS instance; see “Stopping Certificate Management System” on page 110.
2. Open the configuration file (`CMS.cfg`) in a text editor; to locate the file, see “Locating the Configuration File” on page 71.

3. Change the configuration:

- To change the certificate used for authenticating to the Agent Services interface, locate the `agentGateway.https.nickName` parameter and change its value to the nickname of the new SSL server certificate.

For example, if the nickname of the SSL server certificate is `SSLServerCert_agt`, the configuration should look like this:

```
agentGateway.https.nickName=SSLServerCert_agt
<instance_id>
```

- To change the certificate used for authenticating to the end-entity services interface, locate the `eeGateway.https.nickName` parameter and change its value to the nickname of the new SSL server certificate.

For example, if the nickname of the SSL server certificate is `SSLServerCert_ee`, the configuration should look like this:

```
eeGateway.https.nickName=SSLServerCert_ee <instance_id>
```

- To change the certificate used for authenticating to the administration interface, Netscape Console, locate the `radm.https.nickName` parameter and change its value to the nickname of the new SSL server certificate.

For example, if the nickname of the SSL server certificate is `SSLServerCert_admin`, the configuration should look like this:

```
radm.https.nickName=SSLServerCert_admin <instance_id>
```

4. Save your changes.
5. Start the server; see “Starting Certificate Management System” on page 106.

## Getting an SSL Client Certificate for a Subsystem

By default, both Certificate Manager and Registration Manager use their SSL server certificates for SSL client authentication to the publishing directory. If you want these subsystems to use other certificates for authenticating to the publishing directory, you can do so. This section explains how to get an SSL client certificate for a subsystem and how to configure the subsystem to use that certificate for authenticating to the publishing directory.

Getting an SSL client certificate for a subsystem involves the following steps:

- Step 1. Generate a Key Pair for the Subsystem
- Step 2. Generate a Certificate Signing Request for the Key Pair
- Step 3. Submit the CSR to the CA
- Step 4. Ask an Agent to Approve the Request
- Step 5. Install the Certificate in the Internal Database
- Step 6. Configure the Subsystem to Use This Certificate

### Step 1. Generate a Key Pair for the Subsystem

Generate a key pair for the subsystem following the instructions in “Generating a New Key”; see Appendix E, “Key Database Tool” in the online version of this document.

### Step 2. Generate a Certificate Signing Request for the Key Pair

Generate a certificate signing request (CSR) for the key pair following the instructions in “Creating a Certificate Request”; see Appendix D, “Certificate Database Tool” in the online version of this document. Be sure to include your email address in the request so that the CA can deliver the certificate to you.

### Step 3. Submit the CSR to the CA

Submit the CSR to the CA. Because the request needs some modifications, be sure to use the *manual* enrollment process; this way you can tell the person who will process your request about the changes he or she must make before approving the request. For example, if you are submitting the request to a Certificate Manager (CA), be sure to use the *manual* enrollment form; do not use the automated enrollment forms, such as the directory-based server enrollment form. Include your email address in the request so that the CA can deliver the certificate to you.

### Step 4. Ask an Agent to Approve the Request

If you submitted the request to a Certificate Manager (CA), ask its agent to process the request. The agent must make sure that only the SSL Client option for certificate type is selected in the request, and then approve it. (For certificates with no Netscape Certificate Type extensions, the key usage extension must be included with *signing* and *encryption* bits set.)

If you submitted the request to any other CA, you must ask the person managing that CA to make the same changes to the request before approving it.

### Step 5. Install the Certificate in the Internal Database

When you receive the request, follow the instructions in “Adding a Certificate to the Database” (see Appendix D, “Certificate Database Tool” in the online version of this document) and install the certificate in the token you used to generate the key pair.

### Step 6. Configure the Subsystem to Use This Certificate

After you install the certificate, configure the corresponding subsystem to use the new certificate for SSL client authentication to the publishing directory. To configure the Certificate Manager, see “Identifying a Certificate Manager’s Publishing Directory” on page 508. To configure the Registration Manager, see “Identifying a Registration Manager’s Publishing Directory” on page 515.

## Setting Up Cipher Preferences for SSL Communications

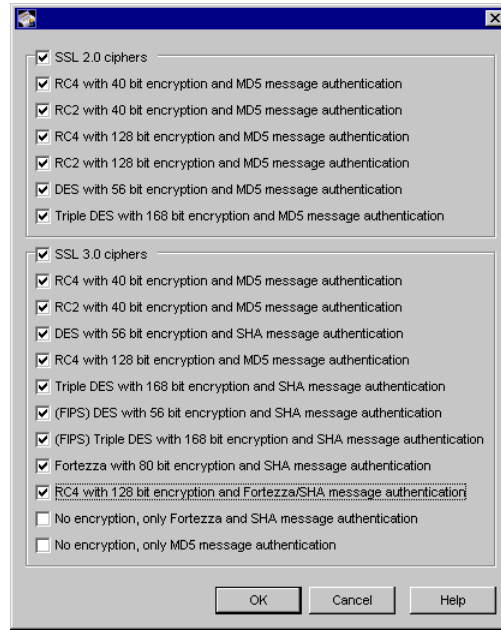
A *cipher* is the algorithm used in encryption. Some ciphers have *stronger* encryption capabilities than others. Generally speaking, the more bits a cipher uses during encryption, the harder it is to decrypt the data.

When a client initiates an SSL connection with Certificate Management System, it lets the server know what ciphers it prefers to use to encrypt information. In any two-way encryption process, both parties must use the same ciphers. A number of ciphers are available; your server needs to be able to use the most popular ones.

### SSL Ciphers Supported in Certificate Management System

Figure 8.1 shows the ciphers supported by Certificate Management System (on the server side). The figure shows SSL 2.0 and 3.0 ciphers supported in the US (or domestic) version of Certificate Management System. The export version of the software shows only those ciphers that are currently permitted by United States law.

Figure 8.1 SSL version 2.0 and 3.0 cipher suites supported (in the US version of the software)



You can choose ciphers from the SSL 2.0 protocol, as well as from SSL 3.0. To specify which ciphers your server can use, check them in the list of ciphers to enable them. Unless you have a compelling reason not to use a specific cipher, you should check them all, except as noted in the warning that follows. For a detailed description of ciphers, see "Ciphers Used with SSL" in Appendix E of *Managing Servers with Netscape Console*.

**Warning** You might not want to check the options that say “No Encryption, only MD5 message authentication” and “No Encryption, only Fortezza and SHA message authentication.” The reason for this is, if no other ciphers are available on the client side, the server will use these and no encryption will occur.

Another reason not to enable all ciphers is to prevent SSL connections with less than optimal encryption. US law prohibits the export of products with 128-bit encryption, so overseas clients might be using only 40-bit encryption, which is not as difficult to crack as 128-bit. Disabling all 40-bit ciphers effectively restricts access to browsers available only in the United States.

Note that international users of Netscape Communicator (with encryption capability restricted to 40-bit encryption) can use Netscape's International Step-Up program to *step up* to stronger encryption, 56-bit, 128-bit, or 168-bit. Step-up refers to the ability of export browsers to establish strong SSL sessions with domestic SSL servers, if they have the appropriate step-up certificates.

For general information on the International Step-Up program and getting step-up certificates, see the information available at the following URL:

```
http://home.netscape.com/products/security/technology/  
128bit.html
```

For information on configuring up your clients and servers for the international step-up feature (after you get the step-up certificates), see the information available at this URL:

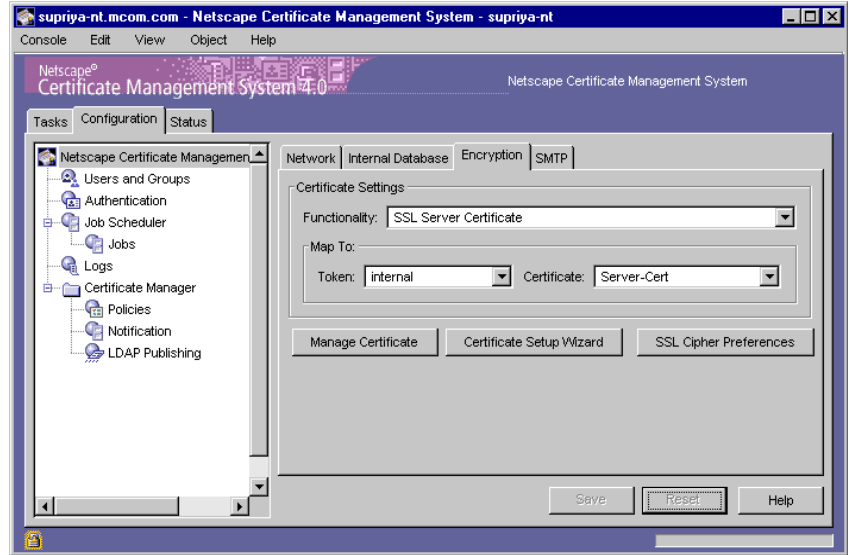
```
http://developer.netscape.com/tech/security/stepup/  
overview.html
```

## Configuring the Server to Use Specific Ciphers

You can set a number of systemwide preferences for SSL by specifying the ciphers that Certificate Management System should recognize and use during SSL communication; the server applies the cipher settings you choose to all the SSL (HTTPS) ports it uses.

To change the cipher settings for a CMS instance:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab, and then in the right pane, click the Encryption tab.



3. Click SSL Cipher Preferences, and choose the appropriate options.

For details, see “Setting Up Cipher Preferences for SSL Communications” on page 228.

4. Click OK.

You are returned to the Encryption tab.

5. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

# Getting New Certificates for the Subsystems

You may need to get new certificates for Certificate Management System. Getting a new certificate means getting a certificate based on a new public and private key pair.

The sections that follow explain how to get new certificates for the Certificate Manager, Registration Manager, and Data Recovery Manager using the Certificate Setup Wizard. Alternatively, you can use the command-line utility called the *Certificate Database Tool*; for details about this utility, see Appendix D, “Certificate Database Tool” in the online version of this document.

Getting a new certificate involves the following steps:

- Step 1. Plan for the New Certificate
- Step 2. Request the New Certificate
- Step 3. Install the New Certificate
- Step 4. Deploy the New Certificate

## Step 1. Plan for the New Certificate

Getting a new certificate for a subsystem requires careful planning. This section provides some guidelines that will help you request and install the new certificate.

### Determine which certificate you want to get

You can get CA signing and SSL server certificates for the Certificate Manager; signing and SSL server certificates for the Registration Manager; and transport and SSL server certificates for the Data Recovery Manager.

- If you want to get a new self-signed CA certificate for a Certificate Manager, you must consider the possible effects on your PKI setup of changing the key pair of the root CA (also called CA key rollover). When you change the root CA key, all certificates that rely on the CA certificate for validation will no longer be validated. Before getting a new self-signed certificate for a Certificate Manager, therefore, you must address issues involved in deploying the new CA certificate across your enterprise.



- If you want to get a new subordinate CA certificate for a Certificate Manager, you must consider the possible effects on your PKI setup of changing the key pair of the subordinate CA. When you change the subordinate CA key, all certificates that rely on the CA certificate for validation will no longer be validated. Before getting a new subordinate certificate, you must plan to address issues involved in deploying the new subordinate CA certificate across your enterprise.
- If you want to get a new signing certificate for a Registration Manager, check whether the Registration Manager has been set up as a trusted manager for any other subsystems—that is, you must identify the subsystems that have been configured to receive requests from this Registration Manager; see “Trusted Managers” on page 141. You will need to replace the existing signing certificate with the new one in all these subsystems.
- If you want to get a new transport certificate for a Data Recovery Manager, you must identify the end-entity interfaces or forms that have been set up for the archival of end users’ encryption private keys; see “How Key Archival Works” on page 627. You will need to replace the existing transport certificate with the new one in all these forms.
- If you want to get a new SSL server certificate for a Certificate Manager or Registration Manager, decide whether you want the subsystem to also use this certificate for SSL client authentication to the publishing directory. If so, you will have to request the certificate with the appropriate extensions set, and after installing the certificate you will have to configure the publishing directory.
- You can get any number of SSL server certificates.

### **Decide on the CA that will sign the certificate**

If you want to get a new self-signed CA certificate, you don’t have to make this decision, because the CA itself signs it. For all other certificates, you must decide on the CA that will sign the certificate.

If you want the certificate to be signed by an internally deployed CA, check to be sure that the CA can issue the certificate you want request.

If you want the certificate to be signed by a public CA, find out the following:

- Does the public CA have a public policy statement? If one is available, read it; it may help you decide whether to request the certificate from this CA.
- Is the public CA's certificate already installed in the trusted CA in the trust database of Certificate Management System? If not, do you want to install it?
- Is the public CA a trusted CA in the trust database of Certificate Management System? If not, do you want to trust it?
- Can the public CA issue the certificate you want to request?
- Does the public CA impose any restrictions on certificates it issues? For example, if you are planning for requesting a subordinate CA certificate for a Certificate Manager, you may want to find out whether the public CA imposes any restrictions on the validity period, volume, or type of certificates your CA can issue. If you are planning for requesting a signing certificate for a Registration Manager, you may want to find out whether the public CA imposes any restrictions on the validity period or the number of certificate requests the Registration Manager can sign using the certificate. If you are planning for requesting a transport certificate for a Data Recovery Manager, you may want to find out whether the public CA imposes any restrictions on the validity period or the number of keys the Data Recovery Manager can archive using the certificate.
- What information does the public CA expects you to provide with the certificate request?
- How long will the public CA take to deliver the certificate, and how will the certificate be delivered to you? (The most common delivery mechanism is by email.)

### **Determine the token for generating the key pair**

Identify the token, internal or external, that you want to use to generate the key pair for the certificate and to store the certificate. If you want to use an existing token, you must know the password that protects the token. If the token is external, make sure that the token is installed properly; see "Installing External Tokens" on page 194.

## **Determine certificate formulation information**

Decide on the subject DN and nickname for the certificate you want generate. Also decide on details such as the key algorithm, key size, extensions, and validity period for the certificate.

## **Step 2. Request the New Certificate**

Once you have all the information, go ahead and request the certificate. The Certificate Setup Wizard built into the CMS window automates the process of requesting certificates used by the Certificate Manager, Registration Manager, and Data Recovery Manager. You can use this wizard to generate a new certificate request and submit the request to any CA for signing. For instructions, see “Using the Wizard to Request a Certificate” on page 200.

## **Step 3. Install the New Certificate**

When you receive the certificate from the CA, you must install it in the token that contains the key pair for this certificate; it must be the token you used to generate the request in Step 2.

The Certificate Setup Wizard automates the process of installing certificates used by the Certificate Manager, Registration Manager, and Data Recovery Manager. You may use this wizard to install the new certificate. For instructions, see “Using the Wizard to Install a Certificate or Certificate Chain” on page 215.

## **Step 4. Deploy the New Certificate**

In this step, follow the instructions appropriate for the certificate you installed:

- If you installed a new CA signing certificate for a Certificate Manager, see “Deploying Certificate Manager’s CA Signing Certificate” on page 236.
- If you installed a new signing certificate for a Registration Manager, see “Deploying Registration Manager’s Signing Certificate” on page 236.
- If you installed a new transport certificate for a Data Recovery Manager, see “Deploying Data Recovery Manager’s Transport Certificate” on page 237.

- If you installed a new SSL server certificate, see “Deploying a Subsystem’s SSL Server Certificate” on page 239.

## Deploying Certificate Manager’s CA Signing Certificate

If you installed a new CA signing certificate, you must deploy it in the PKI environment that depends on this certificate for validation. It is beyond the scope of this book to explain how you should deploy the new CA certificate. You may find it useful to go over some of the deployment issues discussed in the document available at this URL:

<http://help.netscape.com/kb/server/980710-25.html>

## Deploying Registration Manager’s Signing Certificate

If you installed a new Registration Manager signing certificate, you must also install this certificate in the certificate database of all subsystems (Certificate Manager, Registration Manager, and Data Recovery Manager) that trust this Registration Manager.

Here’s what you must do:

- I. Install the new signing certificate in the subsystems’ certificate databases.

Because the Registration Manager uses its signing certificate for SSL client authentication to the subsystems, you must add the new signing certificate to the internal database of all subsystems that have been configured to receive requests from the Registration Manager.

To add the new certificate to a subsystem’s internal database:

- Note the instance ID and host name of the Registration Manager for which you got the new signing certificate; this information will help you to identify the Registration Manager in a subsystem’s list of privileged users.
- Copy the new signing certificate, in base-64 encoded format, to a text file.

- Add the new certificate to the individual subsystem's internal database following the instructions in "Changing a Privileged User's Certificate" on page 178. Repeat this step for all subsystems that receive requests from this Registration Manager.
2. Ensure that the CA that signed the Registration Manager's certificate is in the certificate database of the subsystem.

When a Registration Manager does SSL client authentication using its new certificate, the subsystem, as a part of validating the certificate presented by the Registration Manager, checks its trust database for the CA (certificate) that signed the Registration Manager's new certificate. If the subsystem does not find the CA as a trusted CA in its trust database, it rejects the Registration Manager.

For instructions on checking the trust database of a subsystem, see "Viewing the Certificate Database Contents" on page 247.

- If you don't find the CA certificate, add it to the database as a trusted CA. For instructions on adding a CA certificate to the trust database of a subsystem, see "Installing a New CA Certificate in the Certificate Database" on page 254.
- If you find the CA certificate, verify its trust status. If it is untrusted, change the status to trusted. For instructions on changing the trust setting of a CA certificate, see "Changing the Trust Settings of a CA Certificate" on page 250.

## Deploying Data Recovery Manager's Transport Certificate

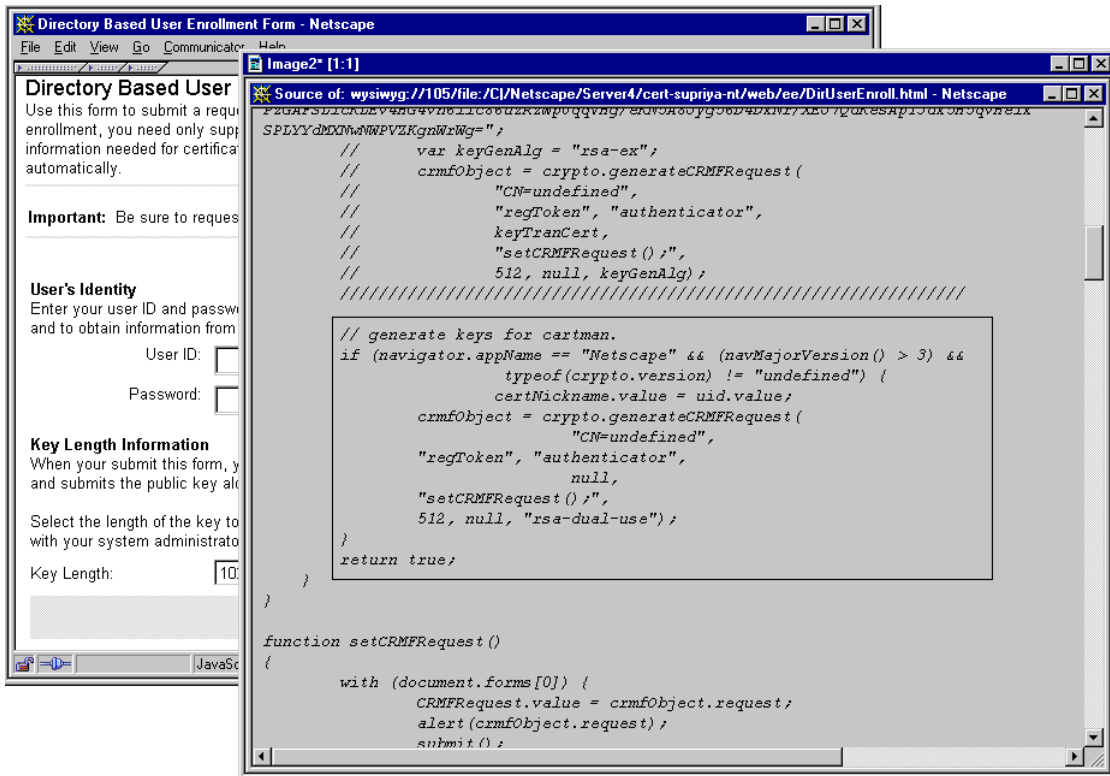
Because clients capable of generating dual key pairs use the transport certificate for encrypting end users' encryption private keys before sending them to the Data Recovery Manager, you must update the appropriate enrollment or key archival page to identify and use the new transport certificate. Otherwise, the Data Recovery Manager will fail to archive users' encryption private keys.

In general, here's what you need to do:

1. Locate the enrollment page that embeds the key archival feature.
2. View the HTML source, and identify the parameter that corresponds to the Data Recovery Manager's transport certificate.

The default enrollment forms for end users embed this feature. Figure 8.2 shows the default directory-based user enrollment form with the transport certificate-related information highlighted. (For more information, see "Step 3. Customize the Certificate Enrollment Form" on page 643.)

Figure 8.2 Data Recovery Manager's transport certificate information in the default enrollment form



3. Replace the current MIME-64 string with the one for the new transport certificate.

To copy the MIME-64 string for the new transport certificate, locate the new transport certificate; the MIME-64 string for the certificate will be listed there.

4. Repeat steps 1 through 3 for any additional enrollment or key archival pages.

## Deploying a Subsystem's SSL Server Certificate

By default, a Certificate Manager and Registration Manager use a single SSL server certificate to do server-side authentication to all the CMS ports. If configured for LDAP publishing, they also use the SSL server certificate for authenticating to the publishing directory. Depending on the purpose for which you requested this certificate, you should configure the server appropriately.

To configure the server to use this certificate for authenticating to one of the clients, see “Configuring the Server to Use Separate SSL Server Certificates” on page 224. To configure the Certificate Manager to use this certificate for authenticating to the publishing directory, see “Identifying a Certificate Manager's Publishing Directory” on page 508. To configure the Registration Manager to use this certificate for authenticating to the publishing directory, see “Identifying a Registration Manager's Publishing Directory” on page 515.

# Renewing Certificates for the Subsystems

All certificates used by Certificate Management System have a validity period beyond which they are not usable, unless the validity period is extended. For Certificate Management System to function properly, you must renew the certificates used by the Certificate Manager, Registration Manager, or Data Recovery Manager before they expire. For example, if you generated these certificates during CMS installation with a validity period of two years, at the end of which they will all expire; you must consider renewing them well in advance, maybe two months in advance.

When you renew a certificate, you get a new certificate with the same public and private key material as the existing certificate, but with a new name or an extended validity period, or both.

The sections that follow explain how to renew certificates for the Certificate Manager, Registration Manager, and Data Recovery Manager using the Certificate Setup Wizard. Alternatively, you use the command-line utility called the *Certificate Database Tool*; for details about this utility, see Appendix D, “Certificate Database Tool” in the online version of this document.

Renewing an existing certificate involves the following:

- Step 1. Plan for Certificate Renewal
- Step 2. Renew the Existing Certificate
- Step 3. Install the Renewed Certificate
- Step 4. Deploy the Renewed Certificate

## Step 1. Plan for Certificate Renewal

Renewing a subsystem’s certificate requires careful planning. This section provides some guidelines that will help you renew the certificate smoothly.

Before renewing a certificate:

- Note the subject DN and nickname of the certificate you want to renew.

If you are planning on renewing the CA signing certificate of a Certificate Manager, make sure that the Certificate Manager has updated your LDAP directory with the most current certificate information (see “Configuring a Certificate Manager for LDAP Publishing” on page 507) and CRL (see “Configuring a Certificate Manager for Publishing CRLs” on page 527).

When you renew its CA signing certificate, the Certificate Manager automatically formulates a new certificate with the same public key and other details from the existing certificate, and publishes the new CA certificate to your LDAP directory.

- Identify the token, internal or external, that contains the keys for the certificate you want to renew. To use an existing token, you must know the password that protects the token. If the token is external, make sure that the token is installed properly; see “Installing External Tokens” on page 194.
- Decide on the validity period of the renewed certificate.



- Decide on the CA that will sign the certificate. If you want the certificate to be signed by a public CA, find out what information you need to provide with the certificate request. If you want the certificate to be signed by an internally deployed CA, check to be sure it can issue the certificate you want to request.
- Find out how long the CA will take to deliver the certificate to you. Make sure the renewed certificate is delivered to you well in advance so that you have a buffer period for installing and testing the renewed certificate, before the current certificate expires.
- Find out how the certificate will be delivered to you; the most common delivery mechanism is email. Make appropriate arrangements to receive the certificate.
- If you want to renew a subordinate CA certificate, plan how you will deploy the renewed CA certificate to end entities that rely on this certificate for validation.
- If you want to renew a root CA certificate, plan how you will deploy the renewed root CA certificate in your enterprise.

## Step 2. Renew the Existing Certificate

Once you have all the information, go ahead and renew the certificate. The Certificate Setup Wizard built into the CMS window automates the process of renewing certificates used by the Certificate Manager, Registration Manager, and Data Recovery Manager. The wizard can generate a certificate request based on the existing key pair and submit the request to a CA for signing. For instructions on using the wizard, see “Using the Wizard to Request a Certificate” on page 200.

**Important** Be sure to use the existing key pair to generate the certificate signing request.

## Step 3. Install the Renewed Certificate

When you receive the renewed certificate from the CA, you must install it in the token that contains the key pair for the certificate; this is the token you used to generate the request in Step 2.

The Certificate Setup Wizard automates the process of installing certificates used by the Certificate Manager, Registration Manager, and Data Recovery Manager. For instructions on using the wizard, see “Using the Wizard to Install a Certificate or Certificate Chain” on page 215.

## Step 4. Deploy the Renewed Certificate

In this step, you must follow the instructions appropriate for the certificate you installed:

- If you installed a renewed CA signing certificate for a Certificate Manager, see “Deploying Certificate Manager’s Renewed CA Signing Certificate” on page 242.
- If you installed a renewed signing certificate for a Registration Manager, see “Deploying Registration Manager’s Renewed Signing Certificate” on page 243.
- If you installed a renewed Data Recovery Manager transport certificate, see “Deploying Data Recovery Manager’s Renewed Transport Certificate” on page 244.
- If you installed a new SSL server certificate, see “Deploying a Subsystem’s Renewed SSL Server Certificate” on page 246.

### Deploying Certificate Manager’s Renewed CA Signing Certificate

If you installed a new CA signing certificate, deploy it in the PKI environment that depends on this certificate for validation. It is beyond the scope of this book to explain how you should deploy the new CA certificate. You may find it useful to go over some of the deployment issues discussed in the document available at this URL:

<http://help.netscape.com/kb/server/980710-25.html>

## Deploying Registration Manager's Renewed Signing Certificate

Here's what you must do:

1. Install the renewed signing certificate in the subsystem's certificate database.

Because the Registration Manager uses its signing certificate for SSL client authentication to the subsystems, you must add the renewed signing certificate to the internal database of all subsystems that have been configured to receive requests from the Registration Manager.

To add the renewed certificate to a subsystem's internal database:

- Note the instance ID and host name of the Registration Manager for which you got the signing certificate; this information will help you to identify the Registration Manager in a subsystem's list of privileged users.
  - Copy the renewed signing certificate, in base-64 encoded format, to a text file.
  - Add the renewed certificate to the individual subsystem's internal database following the instructions in "Changing a Privileged User's Certificate" on page 178. Repeat this step for all subsystems that receive requests from this Registration Manager.
2. Ensure that the CA that signed the Registration Manager's certificate is in the trust database of the subsystem.

When a Registration Manager does SSL client authentication using its renewed certificate, the subsystem, as a part of validating the certificate presented by the Registration Manager, checks its trust database for the CA (certificate) that signed the Registration Manager's renewed certificate. If the subsystem does not find the CA as a trusted CA in its trust database, it rejects the Registration Manager.

For instructions on checking the trust database of a subsystem, see "Viewing the Certificate Database Contents" on page 247.

- If you don't find the CA certificate, add it to the database as a trusted CA. For instructions on adding a CA certificate to the trust database of a subsystem, see “Installing a New CA Certificate in the Certificate Database” on page 254.
- If you find the CA certificate, verify its trust status. If it is untrusted, change the status to trusted. For instructions on changing the trust setting of a CA certificate, see “Changing the Trust Settings of a CA Certificate” on page 250.

## **Deploying Data Recovery Manager's Renewed Transport Certificate**

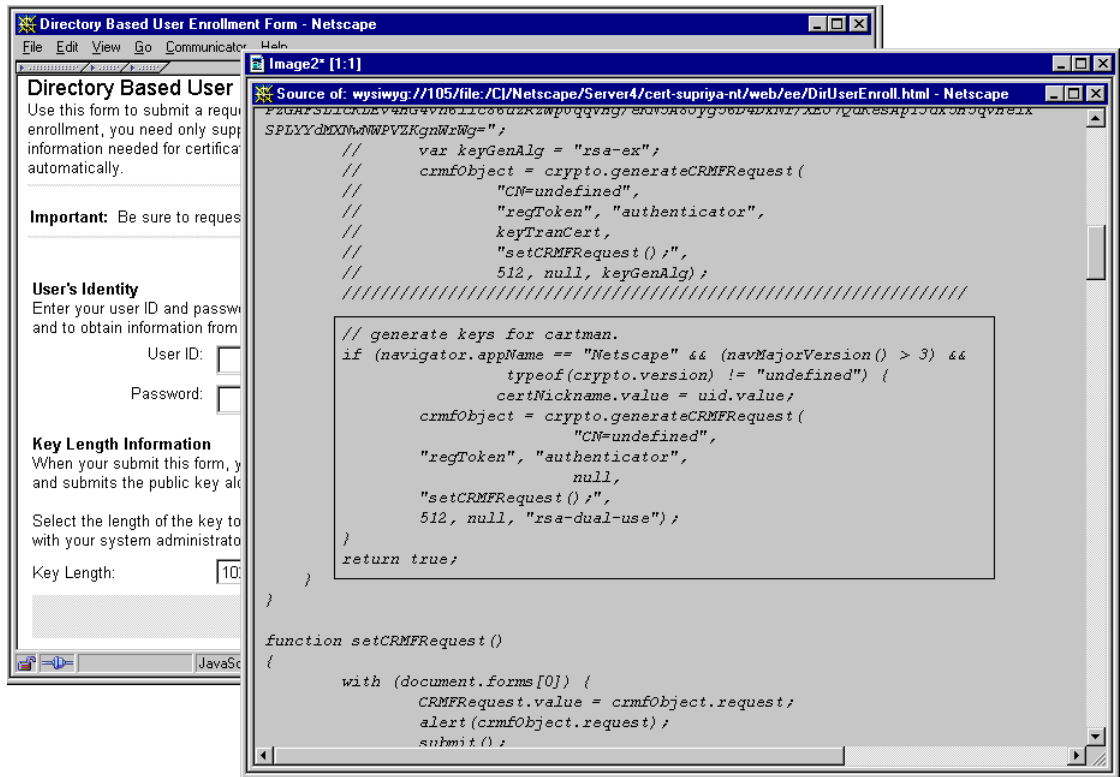
Because clients capable of generating dual key pairs use the transport certificate for encrypting end users' encryption private keys before sending them to the Data Recovery Manager, you must update the appropriate enrollment or key archival page to identify and use the renewed transport certificate. Otherwise, the Data Recovery Manager will fail to archive users' encryption private keys.

In general, here's what you need to do:

1. Locate the page that embeds the key archival feature.
2. View the HTML source, and identify the parameter that corresponds to the Data Recovery Manager's transport certificate.

The default enrollment forms for end users embed this feature. Figure 8.3 shows the default directory-based user enrollment form with the transport certificate-related information highlighted. (For more information, see “Step 3. Customize the Certificate Enrollment Form” on page 643.)

Figure 8.3 Data Recovery Manager's transport certificate information in the default enrollment form



3. Replace the current MIME-64 string with the one for the renewed transport certificate.

To copy the MIME-64 string for the renewed transport certificate, locate the certificate; the MIME-64 string for the certificate will be listed there.

4. Repeat steps 1 through 3 for any additional key archival or enrollment pages.

## Deploying a Subsystem's Renewed SSL Server Certificate

By default, a Certificate Manager and Registration Manager use a single SSL server certificate to do server-side authentication to all the CMS ports. If configured for LDAP publishing, they also use the SSL server certificate for authenticating to the publishing directory. The Certificate Manager, if configured to function as a trusted manager to a Data Recovery Manager, also uses its SSL server certificate for SSL client authentication to the Data Recovery Manager. Depending on the purpose for which the certificate being renewed is used currently, you should configure the server appropriately.

- To configure the server to use this certificate for authenticating to one of the clients, see “Configuring the Server to Use Separate SSL Server Certificates” on page 224.
- To configure the Certificate Manager to use this certificate for authenticating to the publishing directory, see “Identifying a Certificate Manager's Publishing Directory” on page 508.
- To configure the Registration Manager to use this certificate for authenticating to the publishing directory, see “Identifying a Registration Manager's Publishing Directory” on page 515.

## Managing the Certificate Database

Each CMS instance has a certificate database (`cert7.db`), which is maintained in its internal token. This database contains certificates belonging to the subsystems installed in the CMS instance (for details, see “Keys and Certificates for the Main Subsystems” on page 184) and various CA certificates the subsystems use for validating the certificates they receive.

Whether you use an internal token or an external token for generating and storing key pairs, Certificate Management System always maintains its list of trusted and untrusted CA certificates in its internal token.

You may need to add new certificates to the database, remove unwanted certificates from the database, or change the trust settings of CA certificates in the database. This section explains how to view the contents of the certificate

database, delete unwanted certificates, and change the trust settings of CA certificates installed in the database using the CMS window. For information on adding certificates to the database, see “Certificate Setup Wizard” on page 199.

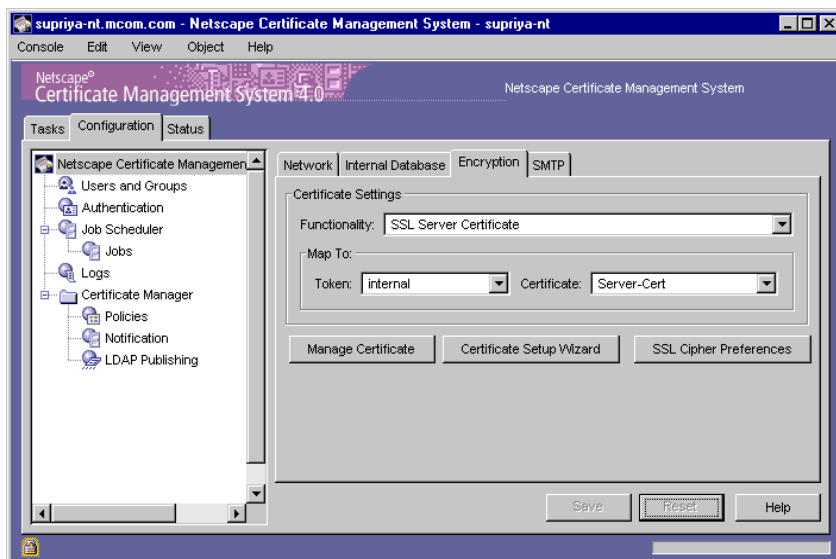
**Note** Certificate Management System provides a command-line utility called `certutil` for managing its certificate database. See Appendix D, “Certificate Database Tool” in the online version of this document.

## Viewing the Certificate Database Contents

Each CMS instance has a certificate database that contains the list of certificates the server uses.

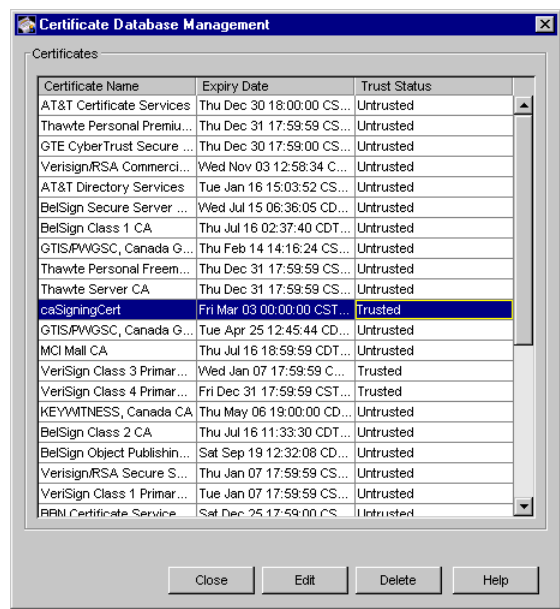
To view the contents of the database:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab, and then in the right pane, click the Encryption tab.



- 3. Click Manage Certificate.

The Certificate Database Management window appears.



The window lists the certificates in a table, with each certificate occupying a row. The CA certificates are listed in alphabetical order. If the database contains multiple CA certificates with the same nickname, they are sorted by their validity periods; the most recently requested certificate is placed at the top.

For each certificate, you see the following information:

**Certificate Name.** Specifies the nickname of the CA certificate.

**Expiry Date.** specifies the date (and time) on which the CA certificate expires.

**Trust Status.** Specifies whether the CA is trusted or untrusted. To change the trust setting, see “Changing the Trust Settings of a CA Certificate” on page 250.



## Deleting a Certificate from the Certificate Database

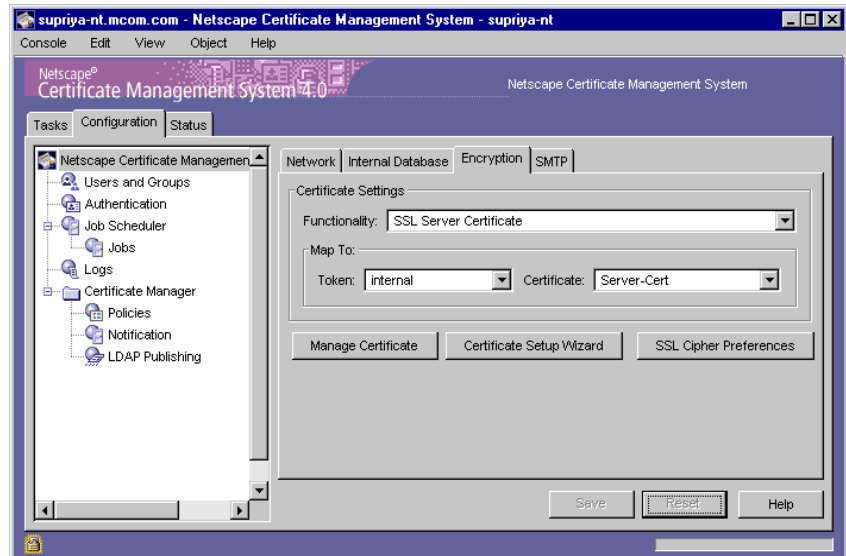
By default, the CMS certificate database includes a few public or third-party CA certificates. As an administrator, you should periodically check the contents of the certificate database and make sure that it doesn't include any unwanted CA certificates. For example, if the database includes CA certificates that you don't ever want to trust in your PKI setup, you should delete them.

Removing unwanted certificates also reduces the size of the certificate database.

**Important** When deleting CA certificates from the certificate database, be careful not to delete the *intermediate CA certificates*, which help a subsystem chain up to the trusted CA certificate. If in doubt, leave the certificates in the database as *untrusted* CA certificates; see “Changing the Trust Settings of a CA Certificate” on page 250.

To delete a CA certificate from the certificate database:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab, and then in the right pane, click the Encryption tab.



3. Click Manage Certificate.

The Certificate Database Management window appears. The window lists all the certificates for the selected instance of Certificate Management System; the list is a table, with each certificate occupying a row.

4. Select the CA certificate you want to delete, and click Delete.
5. When prompted, confirm the delete action.
6. Click Close.

You are returned to the Encryption tab.

7. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

## Changing the Trust Settings of a CA Certificate

Certificate Management System relies on CA certificates in its certificate database for validating certificates it receives during an SSL communication. For example, when a Certificate Manager is authenticating a Registration Manager that has sent a certificate signing requests, the Certificate Manager checks its certificate database to see whether the CA that signed the certificate presented by the Registration Manager is included in the database as a *trusted* CA.

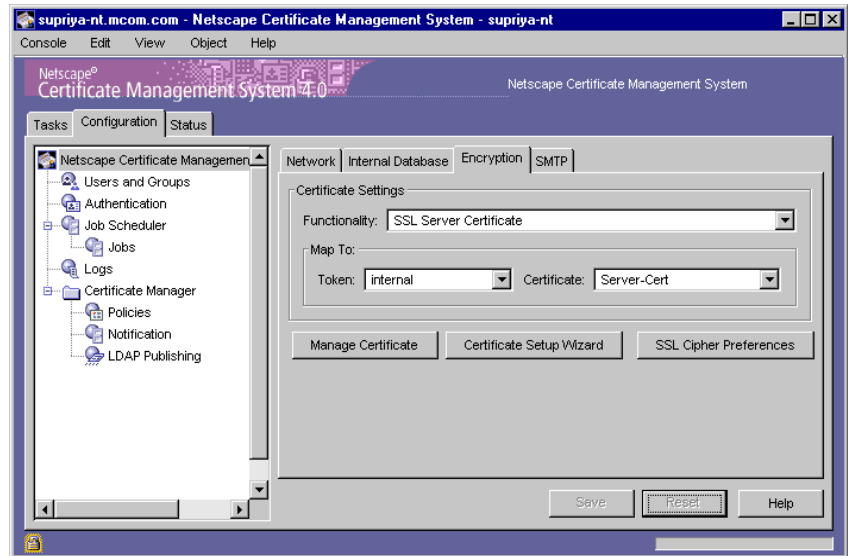
You may need to change the status of a currently trusted CA to untrusted (or vice versa) temporarily or permanently. For example, you may be notified that a CA is experiencing technical difficulty that prevents certificate authentication. By making the CA certificate untrusted, you can prevent entities whose certificates have been signed by that CA from successfully authenticating to Certificate Management System. You can then return the trust option to *trusted* when the CA notifies you that the problem has been resolved.

If you want to untrust a CA permanently, you should consider removing its certificate from the trust database altogether. See “Deleting a Certificate from the Certificate Database” on page 249.

Changing the trust setting changes the trust flag (or bit) in the CA certificate.

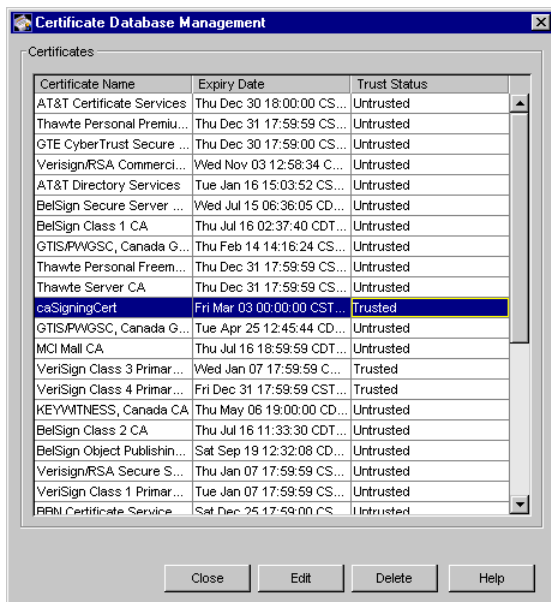
To change the trust setting of a CA certificate:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab, and then in the right pane, click the Encryption tab.



3. Click Manage Certificate.

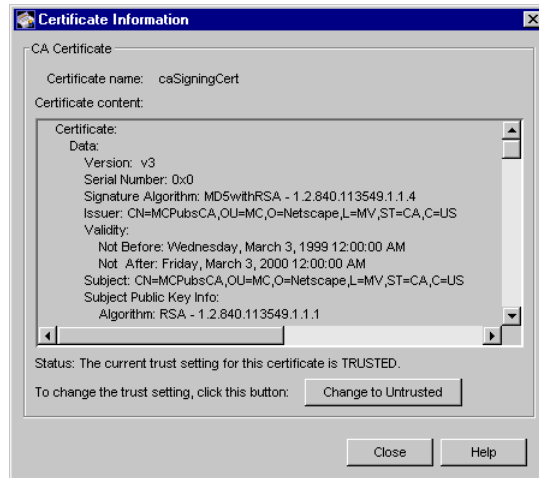
The Certificate Database Management window appears.



The window lists the CA certificates currently installed for the selected CMS instance; the list is a table, with each CA certificate occupying a row.

4. Select the CA certificate whose trust setting you want to modify, and click Edit.

The Certificate Information window appears.



The window shows detailed information about the selected certificate, including serial number, validity period, subject name, issuer name, certificate fingerprint, and trust status.

If the certificate you selected is currently trusted, the window shows a button named “Change to Untrusted.” If it is untrusted, the window shows a button named “Change to Trusted.”

5. Click Change to Untrusted or Change to Trusted, as appropriate.
6. Click Close.

You are returned to the Certificate Database Management window. The certificate now shows a different trust status.

7. Click Close.

You are returned to the Encryption tab.

8. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

## Installing a New CA Certificate in the Certificate Database

You may need to install new trusted CA certificates in the certificate database of a CMS instance. For example, assume that you renewed the signing certificate of a Registration Manager. Also assume that the CA that signed the Registration Manager's certificate is not included in the trust database of the Certificate Manager that has been configured to sign certificate requests from this Registration Manager.

When the Registration Manager attempts to request a service from the Certificate Manager (using the renewed certificate for SSL client authentication), the Certificate Manager fails to authenticate the Registration Manager. This happens because, as a part of validating the certificate presented by the Registration Manager, the Certificate Manager checks its certificate database for the CA that signed the Registration Manager's certificate. The Certificate Manager does not find the CA listed in its trust database as a trusted CA, so it rejects the Registration Manager's service request.

The Certificate Setup Wizard built into the CMS window automates the process of installing trusted CA certificates in the certificate database. For instructions on using the wizard, see "Using the Wizard to Install a Certificate or Certificate Chain" on page 215.

**Important** Be sure to choose the "Other Trusted CAs" option in Step 2 in the wizard process.

## Installing a CA Certificate Chain in the Certificate Database

Any client or server software that supports certificates maintains a collection of trusted CA certificates in its certificate database. These CA certificates determine which other certificates the software can validate—in other words, which issuers of certificates the software can trust. In the simplest case, the software can validate only certificates issued by one of the CAs for which it has a certificate. It's also possible for a trusted CA certificate to be part of a chain of CA certificates, each issued by the CA above it in a certificate hierarchy; for details on certificate hierarchies and certificate chains, see "How CA Certificates Are Used to Establish Trust" in Appendix D of *Managing Servers with Netscape Console*.





# 4

## *Authentication*

*Chapter 9* Introduction to Authentication

*Chapter 10* Using the PIN Generator Tool

*Chapter 11* Configuring Authentication for End Entities

*Chapter 12* Developing Authentication Plug-ins



# Introduction to Authentication

Authentication is the process of verifying the identity of a user that is requesting a service from Netscape Certificate Management System (CMS). More specifically, authentication involves acquiring and verifying the values of the configured attributes of the user. For example, the user might be prompted to log in with a user name and password, and then the server would look in a preconfigured database to verify that the user's password was correct.

Service requests to Certificate Management System come from any of the following users:

- End entities—general users or applications that make certificate issuance, renewal, and revocation requests
- Administrators—privileged users who connect to the server to do system or server administration tasks
- Agents—privileged users who connect to the server to do agent operations

This chapter explains how Certificate Management System identifies and authenticates these users, and it provides details about the various authentication mechanisms supported out of the box. After reading this chapter, you should be able to determine which of the authentication plug-in modules provided out of the box is suitable for your PKI deployment.

This chapter has the following sections:

- Privileged-User Authentication (page 260)
- End-Entity Authentication During Certificate Enrollment (page 265)
- End-Entity Authentication During Certificate Renewal (page 282)
- End-Entity Authentication During Certificate Revocation (page 283)

## Privileged-User Authentication

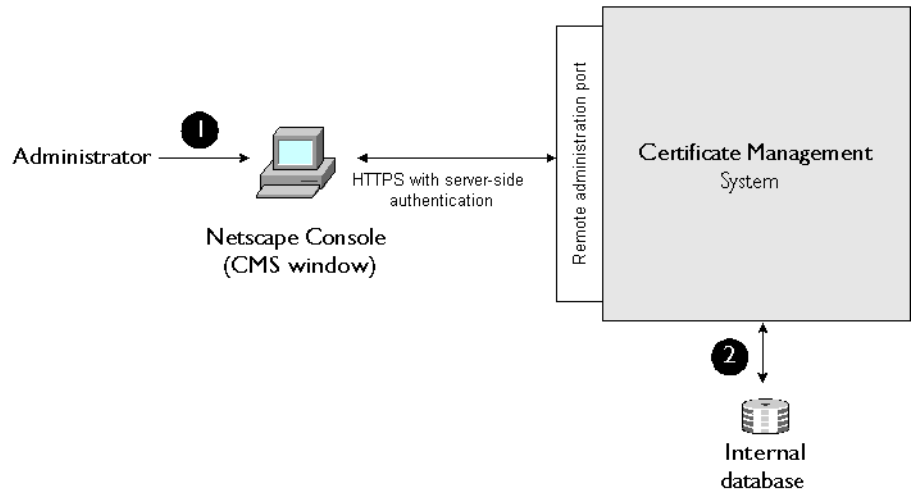
For authenticating privileged users, such as administrators and agents, Certificate Management System uses built-in authentication mechanisms.

### Authentication of Administrators

When an administrator makes an administrative request to Certificate Management System (from the CMS window within Netscape Console or from any command-line tool), the server needs to authenticate the administrator before processing the request. To facilitate this, Certificate Management System supports an authentication mechanism that includes user ID- and password-based authentication from the client and SSL server authentication from the server.

Certificate Management System identifies and authenticates users with *administrator* privileges by checking their user IDs and passwords in its internal database. These are the user IDs and passwords you entered in the internal database when you created these user entries. For details, see “Setting Up Administrators” on page 150.

Figure 9.1 illustrates the authentication process.

Figure 9.1 CMS authentication of a user with *administrator* privileges

These are the steps shown in Figure 9.1:

1. An administrator opens Netscape Console and attempts to log in to the CMS window by entering the user ID and password at the login prompt. The server takes the administrator's user ID and password and binds them to privileged-user entries in its internal database.
2. If the user ID and password bind successfully to a user entry, authentication succeeds; otherwise, it fails.
  - If authentication fails, the server logs an error message and sends a rejection notification. See "Logs" on page 565.
  - If authentication succeeds, the server proceeds to check the user's access rights (based on group memberships) to determine whether the user is authorized to perform the requested operation.

If both authentication and authorization succeed, the server services the request. Otherwise, it rejects the request and logs the reason for the rejection.

**Note** Authentication for administrators is hardcoded; it is not configurable.

## Authentication of Agents

When an agent makes a request to Certificate Management System from the appropriate Agent Services interface, the server needs to authenticate the agent before processing the request. To facilitate this, Certificate Management System supports a certificate-based authentication mechanism.

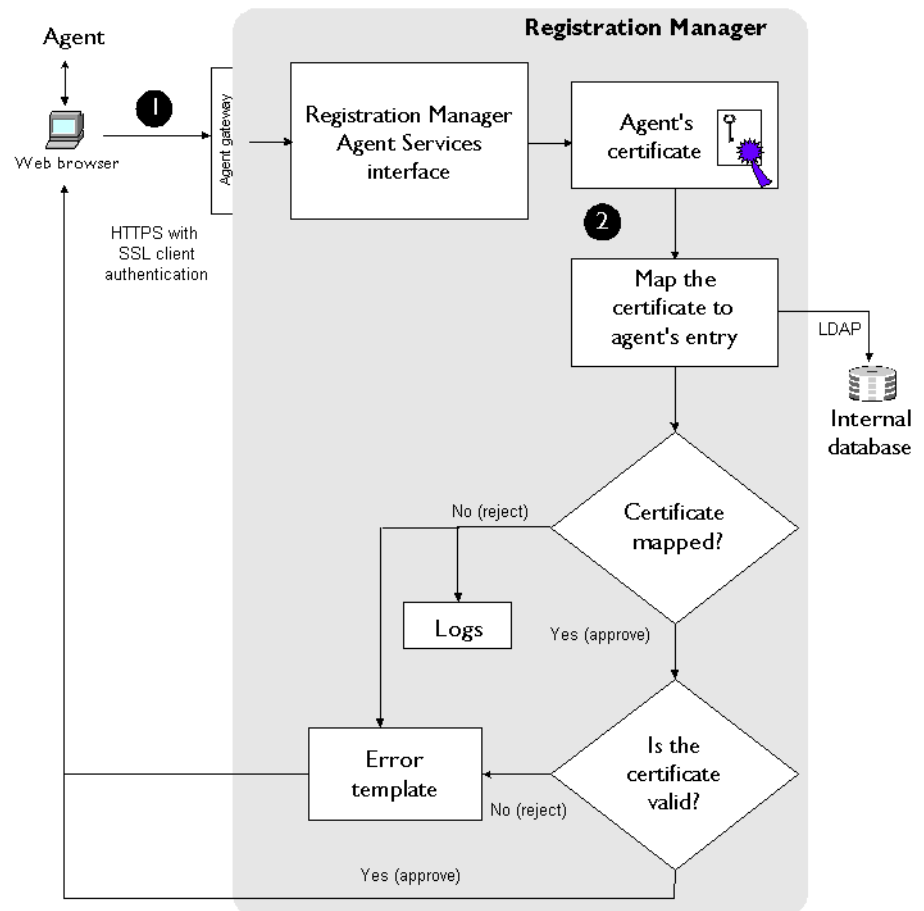
Certificate Management System identifies and authenticates a user with *agent* privileges by checking the user's SSL client certificate in its internal database. The certificates it checks are the ones you imported and stored in the internal database while creating or modifying the user entry. You create agent users for a CMS instance by adding their client certificates into the internal database and associating them with the corresponding users' login information; for details, see "Setting Up Agents" on page 152.

When an agent makes a request to perform a privileged operation, the server requests SSL client authentication from the client that the agent has used to connect to the server. The server then uses the successfully SSL client-authenticated certificate to map to internal user entries for further checks. The server checks the certificate's subject name and issuer name against the list of privileged-user certificates stored in its internal database. If the certificate belongs to a privileged user who is authorized (based on group membership) to perform agent operations, the server allows the user to perform the requested operation. Otherwise, the server rejects the request and logs an appropriate message (for details, see "Logs" on page 565).

**Note** Authentication for agents is hardcoded; it is not configurable.

Figure 9.2 shows how a Registration Manager authenticates and authorizes a Registration Manager agent.

Figure 9.2 Registration Manager authentication of a user with Registration Manager agent privileges



This example shows these steps:

1. An agent opens a web browser and enters the URL to the Registration Manager Agent Services interface hosted by the Registration Manager. The server requests the client for SSL client authentication. The client in turn

prompts the agent to specify the certificate that it should present to the server for authentication. The successfully SSL client authenticated certificate is presented to the Registration Manager.

2. Upon receiving the certificate, the Registration Manager performs the following authentication and authorization process:
  - First, it verifies that the certificate exists in its internal database. Next, it verifies that the certificate is a valid client certificate. If the certificate is valid, the Registration Manager proceeds. Otherwise (for example, if the certificate has expired or been revoked or was signed by an *untrusted* authority), the Registration Manager rejects the request, sends an error message to the agent, and logs a reason for the rejection.

Note that the Registration Manager verifies the revocation status of the agent certificate if it has been issued by the Certificate Manager to which the Registration Manager is connected to; the Certificate Manager keeps a record of all the certificates it has issued and their current status in its internal database. However, if the agent certificate is issued by any other CA, the Registration Manager cannot verify the revocation status of the certificate; it can only verify that the certificate is valid and that it has been issued by a CA that the Registration Manager trusts.

If the internal database contains an invalid certificate for an agent, the server rejects all requests from that agent. For the server to accept requests from that agent, you would have to replace the agent's invalid certificate in the internal database with a valid one. For details on how to do this, see "Changing a Privileged User's Certificate" on page 178.

- The Registration Manager reads the user's subject name (in DN form) and the issuer name from the certificate. This combination is unique. It then finds the login name corresponding to this unique combination in its privileged-users list, which is stored in the internal database. If a login name is associated with the certificate, the Registration Manager proceeds. Otherwise, it rejects the request.

The Registration Manager then checks the group memberships of the login name and the corresponding access rights to determine whether the user is authorized to perform the requested service.

If both authentication and authorization succeed, the Registration Manager services the request. Otherwise, it rejects the request and logs a reason for the rejection.



# End-Entity Authentication During Certificate Enrollment

When an end entity submits a certificate request, a Certificate Manager or Registration Manager's first task is to identify and authenticate the end entity. The server must perform this task before it can register the end entity for certificate issuance. This process includes verifying the end entity's identity based on information the end entity provides and returning enough information about the end entity so that the subject name for the certificate can be constructed.

To cater to a variety of end-entity enrollment scenarios, Certificate Management System supports both manual and automated certificate issuance. Manual issuance is dependent on human agents; it requires that all end-entity certificate requests be approved by agents before the server can process the requests. Automated certificate issuance is dependent on directories that have been populated with end-entity entries so that the server can use them for authentication.

For automated certificate issuance, Certificate Management System supports the following authentication mechanisms out of the box:

- Directory-based authentication using user ID and password
- Directory-based authentication with a personal identification number (PIN) or one-time password that serves as an enrollment or registration token

Because large corporations typically store corporatwide user, group, and network-resource data in LDAP-compliant directories, the directory-based authentication supported out of the box by Certificate Management System is based on LDAP-compliant directories, such as Netscape Directory Server. If you have an LDAP-compliant directory with end-entity data, you can make use of any of the directory-based authentication plug-in modules provided out of the box. In that case, the only information your end entities need to provide during certificate enrollment is their user ID and password for the directory.

If you don't have an LDAP-compliant directory, you can customize end-entity authentication by using your own authentication modules. For details on writing custom modules and adding them to the CMS authentication framework, see "Developing Authentication Plug-ins" on page 329.

Keep in mind that in an automated certificate management setup, Certificate Management System uses the directory-based authentication models only during certificate enrollment. During certificate renewal and revocation processes for end entities, the server relies solely on end entities' current certificates; that is, certificates are renewed or revoked only after successful SSL client authentication.

Certificate Management System also provides HTML forms-based interfaces for all of the authentication mechanisms it supports out of the box. Your end entities can use these forms for certificate enrollment, renewal, and revocation. You can also customize these forms. For details on the HTML forms, see “Agent and End-Entity Interfaces” on page 531.

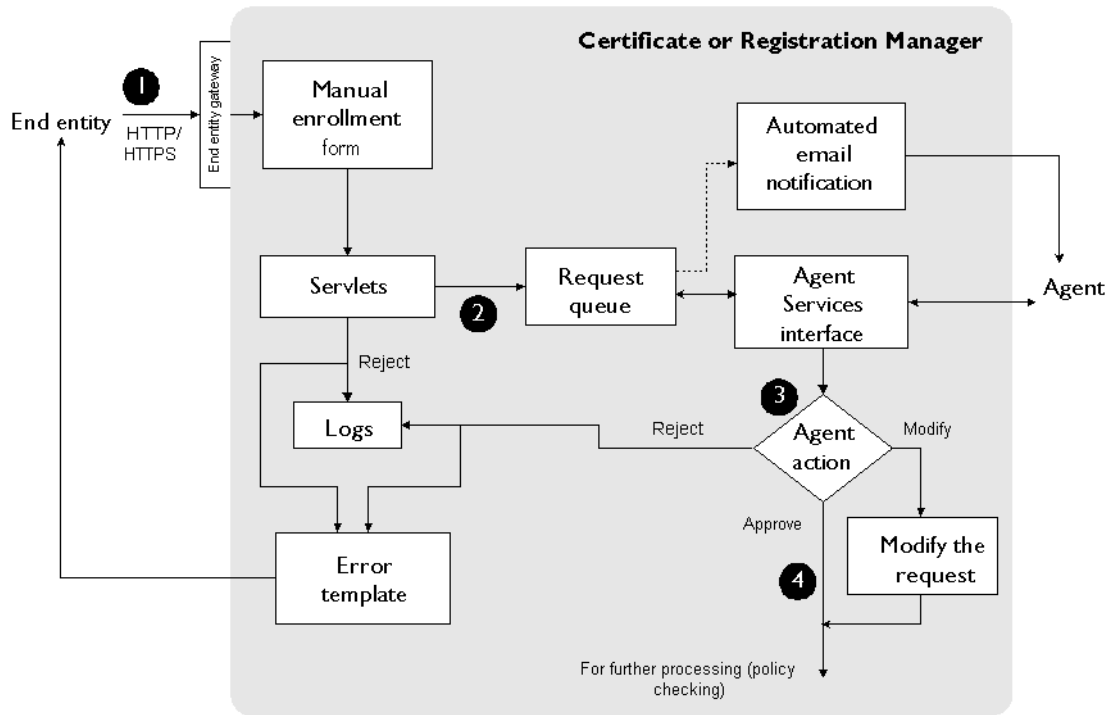
## Manual Authentication

The manual authentication mechanism is suitable for authenticating end entities during certificate enrollment, renewal, and revocation. Use this mechanism for authenticating unprivileged users.

**Note** The manual authentication mechanism is hardwired; you cannot configure it in any other way. This ensures that when the server receives requests that lack authentication credentials, it sends them to the request queue for agent approval. It also means that if you don't configure Certificate Management System for any other authentication mechanism, the server automatically sends all certificate-related requests to a queue where they await agent approval.

Figure 9.3 illustrates how the manual authentication mechanism works during certificate enrollment.

Figure 9.3 Manual authentication of end entities



These are the steps shown in Figure 9.3:

1. In the manual enrollment form, the end entity enters the information required by Certificate Management System to issue a certificate and submits the request to the server.
2. When the server receives the request, it automatically lists the request in a *certificate request queue* for an agent to process.
3. An agent verifies the authenticity of the request.
  - If the request is from a valid end entity, the agent verifies that all the information the end entity has provided in the request is correct, makes required modifications, if any, and approves the certificate request for issuance.

- If the request is from an invalid end entity, the agent rejects the request, which in turn triggers a rejection notification to the end entity.
4. When the server receives the agent-approved request, it subjects it to policy processing; for details, see “Policies” on page 399.
    - If the request fails any of the configured policies, the server rejects the request, logs an error message, and sends a rejection notification to the end entity.
    - If the request passes all the configured policies, the server issues the certificate.

The end entity gets the certificate, which is delivered to the email address provided in the certificate request form.

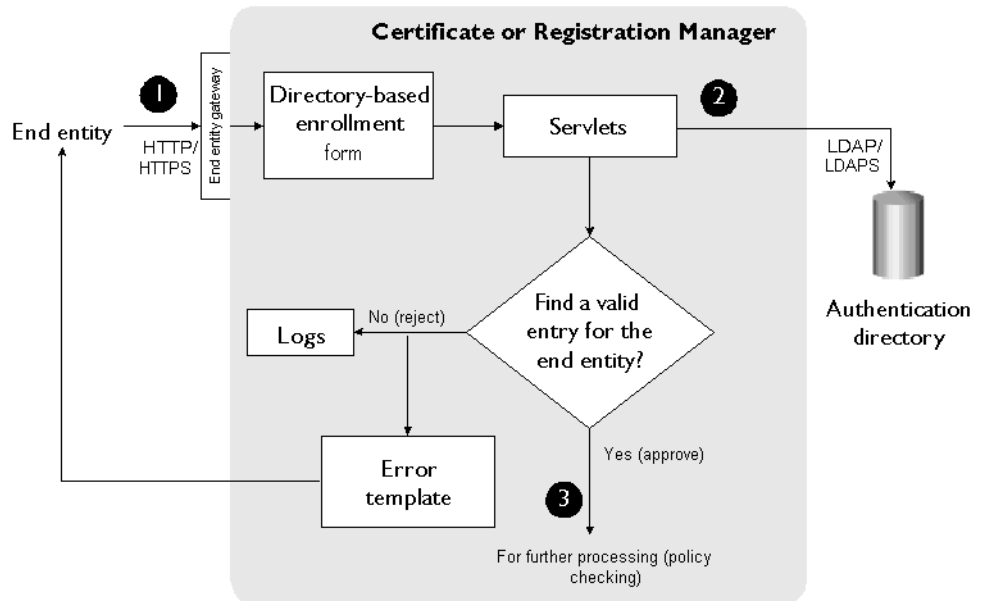
## Directory-Based Authentication

Directory-based authentication is suitable for authenticating end entities during certificate enrollment. Use this mechanism for authenticating unprivileged users in the global LDAP domain.

In this authentication model, the end entities enroll for a certificate by entering their user IDs and passwords for the directory. As part of configuring Certificate Management System for authentication, you specify which directory Certificate Management System must use to authenticate end entities. Once the server successfully authenticates an end entity, it retrieves the rest of the information required to issue a certificate from the directory.

Figure 9.4 illustrates how authentication based on a user ID and password works during certificate enrollment.

Figure 9.4 User ID- and password-based authentication of an end entity



These are the steps shown in Figure 9.4:

1. In the directory-based certificate enrollment form, the end entity enters a user ID and password for the directory and submits the request to Certificate Management System.
2. When the server receives the request, it looks up the directory that is configured for authenticating end entities. The server verifies the authenticity of the end entity by checking the directory entries.
  - If the end entity has a valid entry in the directory, the server retrieves all the information required to construct the subject DN (required for an end entity's certificate).
  - If the end entity does not have a valid entry in the directory, the server rejects the request, logs an error message, and sends a rejection notification to the end entity.

If, for some reason, the directory to which Certificate Management System binds for authenticating the user ID and password is unavailable, the server returns an LDAP error code.

3. Next, the server subjects the certificate request to policy processing; for details, see “Policies” on page 399.
  - If the request fails any of the configured policies, the server rejects the request, logs an error message, and sends a rejection notification to the end entity.
  - If the request passes all the configured policies, the server issues the end entity a certificate.

The end entity gets the certificate, which is delivered to the end entity's email address in the directory.

## Plug-in Module for User ID- and Password-Based Authentication

The plug-in module provided for authenticating end entities based on their user IDs and passwords (for the directory) is identified as follows:

```
com.netscape.certsrv.authentication.UidPwdDirAuthentication
```

- In the configuration file, the implementation for this module is identified as follows; you can find it in the Authentication section:

```
auths.impl.UidPwdDirAuth.class=com.netscape.certsrv.  
authentication.UidPwdDirAuthentication
```

- In the CMS window, the implementation for this module is identified as follows; you can find it in the Authentication Plugin Registration tab:

```
UidPwdDirAuth
```

## Configurable Parameters

Figure 9.5 shows how configurable parameters pertaining to the `UidPwdDirAuth` plug-in implementation are displayed in the CMS window.

Figure 9.5 Parameters and values for the plug-in UidPwdDirAuth

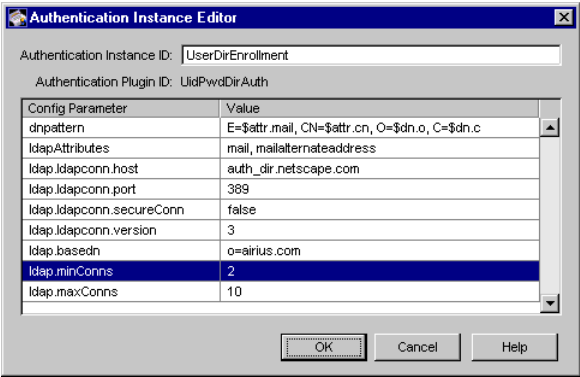


Table 9.1 gives details about each of these parameters and their values.

Table 9.1 Configurable parameters for directory-based authentication and their values

Parameter name	Description
dnpattern	<p>Specifies a string representing a subject name pattern to formulate from the directory attributes and entry DN. The syntax is illustrated in the following example:</p> <p>E=\$attr.mail.1, CN=\$attr.cn, OU=\$dn.ou.2, O=\$dn.o, C=US</p> <p>This sample configuration specifies that the subject name should be formulated as follows:</p> <ul style="list-style-type: none"><li>• E = the first mail LDAP attribute value in the user's entry</li><li>• CN = the (first) cn LDAP attribute value in the user's entry</li><li>• OU = the second ou value in the user's entry DN</li><li>• O = the (first) o value in the user's entry DN</li><li>• C = the string US</li></ul> <p>If this parameter value is empty or not set, the server uses E=\$attr.mail, CN=\$attr.cn, O=\$dn.o, C=\$dn.c as the DN pattern.</p> <p>This default DN pattern works well with Communicator and other browsers. For Communicator, if you leave out E= in end-entity certificates, S/MIME may not work correctly (assuming lack of other extensions in the certificate). Also, if C= and O= are left out, certificate display looks strange in Communicator (when the Display Certificate button is clicked).</p> <p>Permissible values: Any valid DN string composed from standard DN attributes, which must be separated by commas</p> <p>Object type: String</p>



Table 9.1 Configurable parameters for directory-based authentication and their values (Continued)

Parameter name	Description
<code>ldapAttributes</code>	<p>Specifies the list of LDAP attributes that should be considered <i>authentic</i> for the end entity. If specified, the values corresponding to these attributes will be copied from the authentication directory into the authentication token—that is, values retrieved from this parameter can be used by policy modules to form subject names or to make other policy decision (see “Subject Alternate Name Extension Policy” on page 453).</p> <p>Entering values for this parameter is optional.</p> <p>Example: <code>mail, mailalternateaddress</code></p> <p>This sample configuration specifies that <code>mail</code> and <code>mailalternateaddress</code> should be stored in the authentication token.</p> <p>Permissible values: Any valid LDAP attributes, separated by commas</p> <p>Object type: String</p>
<code>ldap.ldapconn.host</code>	<p>Specifies the host name of the authentication directory.</p> <p>Example: <code>auth_dir.netscape.com</code></p> <p>Permissible values: The name must be in this form:  <code>&lt;machine_name&gt;.&lt;your_domain&gt;.&lt;domain&gt;</code></p> <p>Object type: String</p>
<code>ldap.ldapconn.port</code>	<p>Specifies the TCP/IP port at which the authentication directory listens to requests from Certificate Management System.</p> <p>Example: <code>389</code></p> <p>Permissible values: Any valid port number</p> <p>Object type: Integer</p>

Table 9.1 Configurable parameters for directory-based authentication and their values (Continued)

Parameter name	Description
<code>ldap.ldapconn.secureConn</code>	<p>Specifies the type—SSL or non-SSL—of the port at which the authentication directory listens to requests from Certificate Management System.</p> <ul style="list-style-type: none"><li><code>true</code> - specifies HTTPS port.</li><li><code>false</code> - specifies HTTP port.</li></ul> <p>Example: <code>false</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"><li><code>true</code></li><li><code>false</code></li></ul> <p>Object type: String</p>
<code>ldap.ldapconn.version</code>	<p>Specifies the LDAP protocol version.</p> <ul style="list-style-type: none"><li><code>2</code> - specifies LDAP version 2.</li><li><code>3</code> - specifies LDAP version 3.</li></ul> <p>If your authentication directory is based on Netscape Directory Server 1.x, choose LDAP version 2. For Directory Server versions 3.x and later, choose LDAP version 3.</p> <p>Example: <code>2</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"><li><code>2</code></li><li><code>3</code></li></ul> <p>Object type: String</p>
<code>ldap.baseDN</code>	<p>Specifies the base DN for searching the authentication directory.</p> <p>Example: <code>o=airius.com</code></p> <p>Permissible values: Any valid DN string of up to 255 characters</p> <p>Object type: String</p>

Table 9.1 Configurable parameters for directory-based authentication and their values (Continued)

Parameter name	Description
<code>ldap.minConns</code>	<p>Specifies the minimum number of connections permitted to the authentication directory.</p> <p>Example: 2</p> <p>Permissible values: 1 to 3</p> <p>Object type: Integer</p>
<code>ldap.maxConns</code>	<p>Specifies the maximum number of connections permitted to the authentication directory.</p> <p>Example: 10</p> <p>Permissible values: 3 to 10</p> <p>Object type: Integer</p>

## Directory-Based Authentication with PINs

This authentication model is functionally very similar to the directory-based authentication model based on user ID and password, except that for stronger authentication you combine a PIN or one-time password with the end entity's user ID and password. Use this mechanism for authenticating unprivileged users in the global LDAP domain.

In the *directory plus PIN-based authentication model*, the end entities enroll for a certificate by entering their user IDs, passwords, and PINs for the authentication directory. As part of configuring Certificate Management System for authentication, you specify the directory that Certificate Management System must use to authenticate end entities. Once the server successfully authenticates an end entity, it retrieves the rest of the information required to issue a certificate from the directory.

In the authentication model based on user ID, password, and PIN, you have the following choices:

- Authentication without PIN removal

Certificate Management System does not remove the PIN from the directory following end-entity authentication.

- Authentication with PIN removal

Certificate Management System removes the PIN from the directory following end-entity authentication.

**Important** End-entity entries in directories usually do not contain PINs. To be able to use the directory-based authentication with PINs, you must add PINs to the authentication directory first. To do this, follow the information provided in “Using the PIN Generator Tool” on page 285.

## Plug-in Module for User ID-, Password-, and PIN-Based Authentication

The plug-in module provided for authenticating end entities based on their user IDs, passwords, and PINs is identified as follows:

```
com.netscape.certsrv.authentication.UidPwdPinDirAuthentication
```

- In the configuration file, the implementation for this module is identified as follows; you can find it in the Authentication section:

```
auths.impl.UidPwdPinDirAuth.class=com.netscape.certsrv.  
authentication.UidPwdPinDirAuthentication
```

- In the CMS window, the implementation for this module is identified as follows; you can find it in the Authentication Plugin Registration tab:

```
UidPwdPinDirAuth
```

## Configurable Parameters

Figure 9.6 shows how configurable parameters pertaining to the `UidPwdPinDirAuth` plug-in module are displayed in the CMS window.

Figure 9.6 Parameters and values for the plug-in UidPwdPinDirAuth

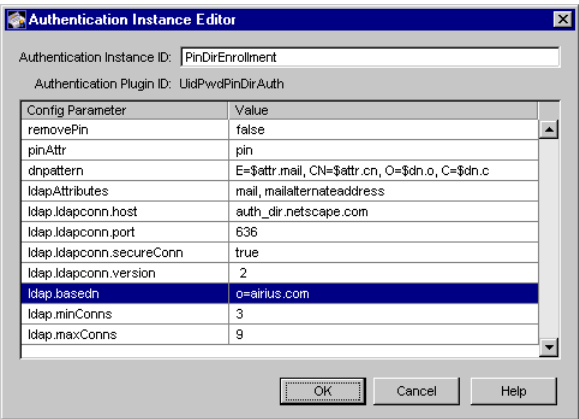


Table 9.2 gives details about each of these parameters.

Table 9.2 Configurable parameters for directory-based authentication with PINs and their values

Parameter name	Description
removePin	<div><p>Specifies whether to remove PINs from the authentication directory (after end entities successfully authenticate).</p><ul style="list-style-type: none"><li>• true - specifies PINs are removed after successful authentication.</li><li>• false - specifies PINs are left in the directory after authentication.</li></ul><p>If you set the value of this parameter to true, you must also specify the values for the removePinBindDN and removePinBindPrompt parameters. Removing PINs from the directory restricts users from enrolling more than once, and thus prevents them from getting more than one certificate.</p><p>Example: true</p><p>Permissible values: Either of the following:</p><ul style="list-style-type: none"><li>• true</li><li>• false</li></ul><p>Object type: String</p></div>

Table 9.2 Configurable parameters for directory-based authentication with PINs and their values (Continued)

Parameter name	Description
<code>ldap.ldapauth. bindDN</code>	<p>Specifies the user entry to bind as when removing PINs from the authentication directory. You need to specify this parameter only if you've set the value of the <code>removePin</code> to true. It is recommended that you create and use a separate user entry that has permission to modify only the PIN attribute in the directory. For example, don't use the directory manager's entry as it has privileges to modify the entire directory content.</p> <p>Example: <code>cn=remove_pin_user</code></p> <p>Permissible values: A valid bind DN</p> <p>Object type: String</p>
<code>ldap.ldapauth. bindPWPrompt</code>	<p>Specifies the text used for constructing the password prompt for entering the password associated with the DN specified by the <code>ldap.ldapauthbindDN</code> parameter. You need to specify this parameter only if you've set the value of the <code>removePin</code> to true.</p> <p>If you configure the authentication instance to remove PINs from the directory, you are required to restart the server from the command line. When you do that, you will be prompted to enter the password associated with the <code>ldap.ldapauthbindDN</code> parameter. The prompt is constructed using the value you enter in this field. For example, if you enter <code>Remove PIN Bind DN</code> as the value, the resulting prompt would be "Enter password for <i>Remove PIN Bind DN</i>".</p> <p>Note that the server prompts for this password when you restart the server the first time after you configure the PIN-based authentication instance. The server then stores the password in the single sign-on password cache and uses it for subsequent start ups (see "Required Start-up Information" on page 106).</p> <p>Permissible values: As applicable</p> <p>Object type: String</p>

Table 9.2 Configurable parameters for directory-based authentication with PINs and their values (Continued)

Parameter name	Description
<code>pinAttr</code>	<p>Specifies the authentication directory attribute for PINs. If you used the PIN Generator, the attribute is specified by the value of the <code>objectclass</code> parameter; the default value for this parameter is <code>pin</code>. For details, see “Arguments” on page 286.</p> <p>Example: <code>pin</code></p> <p>Permissible values: Any valid attribute name</p> <p>Object type: String</p>
<code>dnpattern</code>	<p>Specifies a string representing a subject name pattern to formulate from the directory attributes and entry DN. The syntax is illustrated in the following example:</p> <p><code>E=\$attr.mail.1, CN=\$attr.cn, OU=\$dn.ou.2, O=\$dn.o, C=US</code></p> <p>This sample configuration specifies that the subject name should be formulated as follows:</p> <ul style="list-style-type: none"><li>• <code>E</code> = the first <code>mail</code> LDAP attribute value in the user's entry</li><li>• <code>CN</code> = the (first) <code>cn</code> LDAP attribute value in the user's entry</li><li>• <code>OU</code> = the second <code>ou</code> value in the user's entry DN</li><li>• <code>O</code> = the (first) <code>o</code> value in the user's entry DN</li><li>• <code>C</code> = the string <code>US</code></li></ul> <p>If this parameter value is empty or not set, the server uses <code>E=\$attr.mail, CN=\$attr.cn, O=\$dn.o, C=\$dn.c</code> as the DN pattern.</p> <p>This default DN pattern works well with Communicator and other browsers. For Communicator, if you leave out <code>E=</code> in end-entity certificates, S/MIME may not work correctly (assuming lack of other extensions in the certificate). Also, if <code>C=</code> and <code>O=</code> are left out, certificate display looks strange in Communicator (when the Display Certificate button is clicked).</p> <p>Permissible values: Any valid DN string composed from standard DN attributes, which must be separated by commas</p> <p>Object type: String</p>

Table 9.2 Configurable parameters for directory-based authentication with PINs and their values (Continued)

Parameter name	Description
<code>ldapAttributes</code>	<p>Specifies the list of LDAP attributes that should be considered <i>authentic</i> for the end entity. If specified, the values corresponding to these attributes will be copied from the authentication directory into the authentication token—that is, values retrieved from this parameter can be used by policy modules to form subject names or to make other policy decisions.</p> <p>Entering values for this parameter is optional.</p> <p>Example: <code>mail, mailalternateaddress</code></p> <p>This sample configuration specifies that <code>mail</code> and <code>mailalternateaddress</code> should be stored in the authentication token.</p> <p>Permissible values: Any valid LDAP attributes, separated by commas</p> <p>Object type: String</p>
<code>ldap.ldapconn.host</code>	<p>Specifies the host name of the authentication directory.</p> <p>Example: <code>auth_dir.netscape.com</code></p> <p>Permissible values: The name must be in this form:  <code>&lt;machine_name&gt;.&lt;your_domain&gt;.&lt;domain&gt;</code></p> <p>Object type: String</p>
<code>ldap.ldapconn.port</code>	<p>Specifies the TCP/IP port at which the authentication directory listens to requests from Certificate Management System.</p> <p>Example: <code>636</code></p> <p>Permissible values: Any valid port number</p> <p>Object type: Integer</p>



Table 9.2 Configurable parameters for directory-based authentication with PINs and their values (Continued)

Parameter name	Description
<code>ldap.ldapconn.secureConn</code>	<p>Specifies the type—SSL or non-SSL—of the port at which the authentication directory listens to requests from Certificate Management System.</p> <ul style="list-style-type: none"> <li>• <code>true</code> - specifies HTTPS port.</li> <li>• <code>false</code> - specifies HTTP port.</li> </ul> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li>• <code>true</code></li> <li>• <code>false</code></li> </ul> <p>Object type: String</p>
<code>ldap.ldapconn.version</code>	<p>Specifies the LDAP protocol version.</p> <ul style="list-style-type: none"> <li>• <code>2</code> - specifies LDAP version 2.</li> <li>• <code>3</code> - specifies LDAP version 3.</li> </ul> <p>If your authentication directory is based on Netscape Directory Server 1.x, choose LDAP version 2. For Directory Server versions 3.x and later, choose LDAP version 3.</p> <p>Example: <code>2</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li>• <code>2</code></li> <li>• <code>3</code></li> </ul> <p>Object type: Integer</p>
<code>ldap.baseDN</code>	<p>Specifies the base DN for searching the authentication directory.</p> <p>Example: <code>o=airius.com</code></p> <p>Permissible values: Any valid DN string of up to 255 characters</p> <p>Object type: String</p>

Table 9.2 Configurable parameters for directory-based authentication with PINs and their values (Continued)

Parameter name	Description
<code>ldap.minConns</code>	<p>Specifies the minimum number of connections permitted to the authentication directory.</p> <p>Example: 3</p> <p>Permissible values: 1 to 3</p> <p>Object type: Integer</p>
<code>ldap.maxConns</code>	<p>Specifies the maximum number of connections permitted to the authentication directory.</p> <p>Example: 9</p> <p>Permissible values: 3 to 10</p> <p>Object type: Integer</p>

# End-Entity Authentication During Certificate Renewal

When an end entity submits a certificate renewal request, the first step in the renewal process is for the Certificate Manager or Registration Manager to identify and authenticate the end entity. This includes verifying the status of the end entity's current certificate; either valid or expired is acceptable for renewal, but not revoked.

Certificate Management System verifies the authenticity of a certificate renewal request by mapping the subject name in the certificate being presented for renewal to certificates in its internal database. The server renews the certificate only if the subject name maps successfully to a certificate in its internal database.

Here are a few things to keep in mind about certificate renewal:

- The certificate being presented by the end entity for renewal must be issued by Certificate Management System.
- If the renewal request is processed by a Registration Manager, the end-entity certificate presented must be issued by a Certificate Manager that the Registration Manager knows; the Registration Manager forwards certificate requests to this Certificate Manager for signing.
- The certificate being presented by the end entity for renewal must be currently valid or must have expired; it cannot have been revoked.
- If the renewal lead time does not permit renewing, the server rejects the renewal request.
- If the certificate being presented by the end entity has already been renewed, the server displays the URL for downloading the certificate. This situation may occur if the end entity forgets to download the renewed certificate. It can also happen if the end entity maintains two identical certificate databases on two machines, renews the certificate from one machine, and then tries to renew the same certificate from the other machine.

## End-Entity Authentication During Certificate Revocation

When an end entity submits a certificate revocation request, the first step in the revocation process is for the Certificate Manager or Registration Manager to identify and authenticate the end entity.

When an end user attempts to revoke a certificate, Certificate Management System verifies that the user is attempting to revoke his or her own certificate, not a certificate belonging to someone else. The server also makes sure that the user can revoke only certificates that contain the same subject name as the one in the certificate presented for authentication. After successful authentication, the server lists all certificates belonging to that user. The user can then select the certificate to be revoked or revoke all certificates in the list. The user can also specify additional details, such as the date of revocation and revocation reason for each certificate or for the list as a whole.

Here are a few things to keep in mind about certificate revocation:

- The certificate being presented by the user for revocation must be issued by Certificate Management System. The server verifies the authenticity of a revocation request by mapping the subject name in the certificate being presented for revocation to certificates in its internal database. The server revokes the certificate only if the certificate maps successfully to a valid or expired certificate in its internal database.
- If the revocation request is processed by a Registration Manager, the user certificate presented for authentication must be issued by a Certificate Manager that the Registration Manager knows; the Registration Manager forwards certificate requests to this Certificate Manager for signing.
- The certificate being presented by the user for revocation must be currently valid or must have expired; it cannot have been already revoked.

## Using the PIN Generator Tool

For Netscape Certificate Management System (CMS) to use the plug-in module for *directory-based authentication with PINs*, your authentication directory must contain unique PINs for each end entity to whom you intend to issue a certificate. To aid you in generating PINs for end-entity entries in a directory, Certificate Management System provides a command-line tool called the *PIN Generator*. This tool allows you to generate unique PINs for entries in an LDAP-compliant user directory. The tool stores these PINs (as hashed values) in the same directory against the corresponding user entries, and it copies the PINs to a text file, from which you can deliver the PINs to end entities by an appropriate, secure means.

This chapter explains how to use the PIN Generator. The chapter has the following sections:

- Locating the PIN Generator Tool (page 286)
- The setpin Command (page 286)
- How the Tool Works (page 292)
- Generating PINs (page 299)
- Delivering PINs to End Entities (page 307)

## Locating the PIN Generator Tool

You can find the PIN Generator at this location:

```
<server_root>/bin/cert/tools/setpin.exe
```

<server\_root> is the directory where the CMS binaries are kept. You first specified this directory during installation.

## The setpin Command

You run the PIN Generator by entering the `setpin` command and its arguments in a command shell and monitoring the output in the shell window. This section gives the syntax for the `setpin` command and its arguments. For instructions on generating PINs and storing them in your authentication directory, see “Generating PINs” on page 299.

## Command-Line Syntax

Use the following command in a Unix or DOS command shell:

```
setpin [arguments]
```

### Arguments

The `setpin` command takes the following arguments and options:

```
setpin  
  
[host=<host_name> [port=<port_number>]]  
  
[certdb=<path> nickname=<certificate_nickname> |  
"binddn=<user_id>" bindpw=<bind_password> [SSL=yes | no]]  
  
[objectclass=<objectclass_to_add>]  
  
[attribute=<attribute_name_for_pins>]  
  
["filter=<LDAP_search_filter>"]  
  
[input=<file_name>]
```

```
[length=<PIN_length> | minlength=<minimum_PIN_length>
maxlength=<maximum_PIN_length>]

[gen=RNG-alpha | RNG-alphanum | RNG=printableascii]

[case=upperonly]

[hash=sha1 | md5 | none]

[output=<file_name>]

[clobber]

[write]

[saltattribute=<LDAP_attribute_to_use_for_salt_creation>]

[debug]
```

A description for each argument follows:

- [host=<host\_name> [port=<port\_number>]]

<host\_name> specifies the LDAP directory to connect to.

<port\_number> specifies the TCP/IP port to bind to; the default port number is the default LDAP port, 389.

- [certdb=<path> nickname=<certificate\_nickname> |  
"binddn=<user\_id>" bindpw=<bind\_password> [SSL=yes | no]]

Use this argument to specify how the tool should connect to the directory: whether to use basic authentication, SSL, or SSL with client authentication. By default, SSL is not used (SSL=no).

- If you want the tool to use basic authentication, you must turn off SSL (SSL=no) and enter the bind DN and password information. Do not enter values for the remaining options.
- If you want the tool to use SSL, you must turn on SSL (SSL=yes) and enter the bind DN and password information. Do not enter values for the remaining options.

- If you want the tool to use SSL with client authentication, you must turn on SSL (`SSL=yes`) and enter the nickname of the certificate to use for SSL client authentication and the path to the certificate database that contains this certificate. You don't have to provide the bind DN and password.

`<path>` specifies the path to the certificate database containing the client authentication certificate to use for SSL client authentication. If you provide the path to the certificate database (`cert7.db` in Netscape Directory Server), it is assumed that the LDAP directory has been set up for SSL-based access. You must also specify the certificate for SSL client authentication to the directory.

`<certificate_nickname>` specifies the nickname of the certificate to use for SSL client authentication to the directory. The tool looks for this certificate in the database specified by the `path` parameter value. If you want the tool to use a client certificate residing on a token or smart card to access the directory, prefix the nickname with the word *module* (`module:nickname`); then a PKCS #11 module will be used.

`<user_id>` specifies the user ID that has read and write access to the LDAP directory; the PIN Generator binds to the directory as this user.

`<bind_password>` specifies the password for the user ID that has read and write access to the LDAP directory.

If the bind password is not given at the command line, the tool prompts for it.

- `[objectclass=<objectclass_to_add>]`

Use this argument to specify the object class, if any, the tool should add to the authentication directory.

- `[attribute=<attribute_name_for_pins>]`

Use this argument to specify the authentication directory attribute to which PINs should be published. If you don't specify an attribute, it defaults to `pin`, the new attribute added to the authentication directory schema; see "Step 2. Update the Directory Schema" on page 300.



- [`filter=<LDAP_search_filter>`"]

Use this argument to filter those DNs in the directory for which the tool should generate PINs. For information on how to specify filters, see the information available at this URL:

<http://developer.netscape.com/docs/manuals/dirsdk/capi/search.htm>

- [`input=<file_name>`]

Use this argument to specify the name of the file that contains the list of DNs to process. Using this argument is optional. If you do, the tool compares the filtered DNs to the ones specified by the input file and generates PINs for only those DNs that are also in the file.

- [`length=<PIN_length> | minlength=<minimum_PIN_length> maxlength=<maximum_PIN_length>`]

Use this argument to specify the exact number or a range of characters that a PIN must contain. The PINs can be either a fixed length or generated to be between two values (x,y) inclusive (x,y>0).

`<PIN_length>` specifies the exact length for the PINs. For example, if you want PIN length to be eight characters, enter 8. PIN length must be an integer greater than zero.

`<minimum_PIN_length>` specifies the minimum length for the PINs. For example, if you want PIN length to be at least six characters, enter 6.

`<maximum_PIN_length>` specifies the maximum length for the PINs. For example, if you want PIN length to be nine characters at the most, enter 9.

- [`gen=RNG-alpha | RNG-alphanum | RNG-printableascii`]

Use this argument to specify the type of characters for PINs. The characters in the password can be constructed out of alphabetic characters (RNG-alpha), alphanumeric characters (RNG-alphanum), or any printable ASCII characters (printableascii).

- [case=upperonly]

Use this argument with the `gen` parameter. If you do, the case for all alphabetic characters is fixed to uppercase only; otherwise, the case is mixed. Restricting alphabetic characters to uppercase reduces the overall combinations for the password space significantly.

- [hash=sha1 | md5 | none]

Use this argument to specify the message digest algorithm the tool should use to hash the PINs before storing them in the authentication directory. If you want to store PINs as SHA-1 or MD5 hashed values in the directory, be sure to specify an output file for storing PINs in plain text. You will need the PINs in plain text for delivering them to end entities.

sha1 produces a 160-bit message digest. This option is used by default.

md5 produces a 128-bit message digest.

none does not hash the PINs.

- [output=<file\_name>]

Use this argument to specify the absolute path to the file to which the tool should write the PINs as it generates them; this is the file to which the tool will capture the output.

If you don't specify a filename, the tool will write the output to the standard output. In any case, all the error messages will be directed to the standard error.

- [clobber]

Use this argument to specify whether the tool should overwrite preexisting PINs, if any, associated with a DN (user). If specified, the tool overwrites the existing PINs with the one it generates. Otherwise, it leaves the existing PINs as they are.

- [write]

Use this argument to specify whether the tool should write PINs to the directory. If specified, the tool writes PINs (as it generates) to the directory. Otherwise, the tool does not make any changes to the directory.

For example, if you want to check PINs—that the PINs are being given to the correct users and that they are conforming to the length and character-set restrictions—before updating the directory, do not specify this option. You can check the PINs before updating the directory by looking at the output file; for details, see “Output File” on page 296.

- [saltattribute=<LDAP\_attribute\_to\_use\_for\_salt\_creation>]

Use this argument to specify the LDAP attribute the tool should use for salt creation. If you specify an attribute, the tool integrates the corresponding value of the attribute with each PIN, and hashes the resulting string with the hash routine specified in the hash argument.

If you don't specify this argument, the DN of the user is used. For details, see “How PINs Are Stored in the Directory” on page 298.

- [debug]

Use this argument to specify whether the tool should write debugging information (to the standard error). If `debug=attrs` is specified, the tool writes much more information about each entry in the directory.

## Example

The following command generates PINs for all entries that have the `CN` attribute (in their distinguished name) defined in an LDAP directory named `laiking` that is listening at port 19000. The PIN Generator binds to the directory as user `DirectoryManager` and starts searching the directory from the node `dn=o=airius.com` in the directory tree. The tool overwrites the existing PINs, if any, with the new ones.

```
setpin host=lailing port=19000 "binddn=CN=directoryManager"
bindpw=password "filter=(cn=*)" basedn=o=airius.com clobber
write
```

## How the Tool Works

The Pin Generator allows you to generate PINs for user entries in an LDAP-compliant directory and update the directory with these PINs. To run the `setpin` command, you need at a minimum to specify the following:

- The host name (`host`) and port number (`port`) of the LDAP server
- The bind DN (`binddn`) and password (`bindpw`)
- An LDAP filter (`filter`) for filtering out the user entries that require PINs

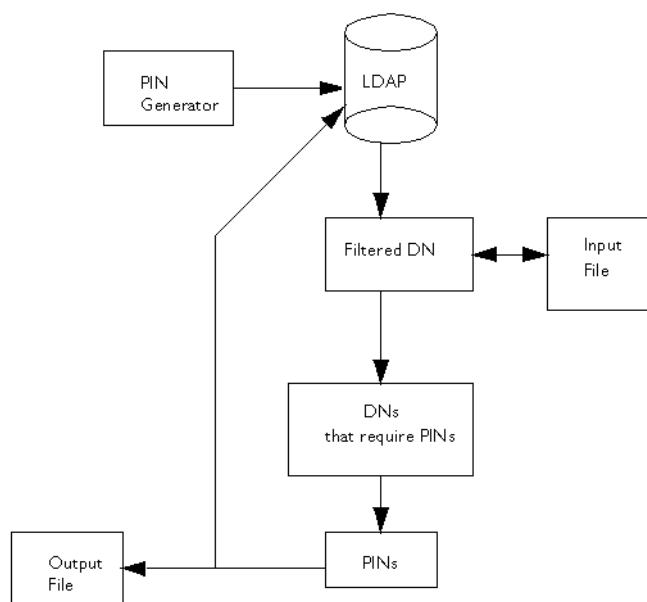
For example:

```
setpin host=laiking port=19000 "binddn=CN=Directory Manager"  
bindpw=netscape "filter=(ou=employees)" basedn=o=airius.com
```

This command, if run, will query the directory for all the entries that match the filter criteria, which in this case is all users belonging to an organizational unit (`ou`) called `employees`. For each entry matching the filter, information is printed out to standard error. Additionally, to the standard output or the file named in `output`; see “Output File” on page 296.

You can also provide the tool with an input argument using the `input` option. The argument must be in the form of an ASCII file of pre-prepared DN and PINs (see Figure 10.1). Note that the input file isn't a substitute for the LDAP directory entries; the filter attribute must still be provided. If an input file is provided, the tool updates only those filtered attributes that match the ones in the input file. For more information about the input file, see “Input File” on page 295.

Figure 10.1 Using an input and output file for the PIN-generation process



Examples of output follow:

Processing: cn=QA Managers,ou=employees,o=airius.com

Adding new pin/password

dn:cn=QA Managers,ou=employees,o=airius.com

pin:ldWynV

status:notwritten

Processing: cn=PD Managers,ou=employees,o=airius.com

Adding new pin/password

dn:cn=PD Managers,ou=employees,o=airius.com

pin:G69uV7

status:notwritten

Because the PIN Generator makes a lot of changes to your directory, it is important that you specify the correct filter; otherwise, you may change the wrong entries. As a safeguard, a `write` option is provided that you use to enable writing to the directory after you verify the output; the tool doesn't make any changes to the directory until you specify the `write` option on the command line.

The output also contains the status of each entry in the directory. It can be one of the values specified in Table 10.1.

**Table 10.1 PIN Generator status**

Exit code	Description
<code>notwritten</code>	Specifies that the PINs were not written to the directory, because the <code>write</code> option was not specified on the command line.
<code>writfailed</code>	Specifies that the tool made an attempt to modify the directory, but the write operation was unsuccessful.
<code>added</code>	Specifies that the tool added the new PIN to directory successfully.
<code>replaced</code>	Specifies that the tool replaced an old PIN with a new one (the <code>clobber</code> option was specified).
<code>notreplaced</code>	Specifies that the tool did not replace the old PIN with a new one (the <code>clobber</code> option was not specified).

If a PIN already exists for a user, it will by default not be changed if you run the `setpin` command a second time. This is so that you can generate PINs for new users without overwriting PINs for users who have previously been notified of their PINs. If you want to overwrite a PIN, you should use the `clobber` option.

Once you are sure that the filter is matching the right users, you should run the `setpin` command again with the `write` option, and with `output` set to the name of the file to capture the unhashed PINs. This output file is in the same format as the input file. For details about the output file, see “Output File” “Output File” on page 296.

# Input File

The PIN Generator can receive a list of DN's to modify in a text file specified by the `input=<file_name>` argument. If you specify an input file, the tool compares the DN's it filtered from the LDAP directory with the ones in the input file, and updates only those DN's that matched the ones in the input file.

The purpose of the input file is multifold. It enables you to provide the Pin Generator with an exact list of DN's to modify. Via the input file, you can also provide the PIN Generator with PIN's (in *plain text* format) for all DN's or for specific DN's.

The following examples explain why you might want to use the input file:

- Assume that you have set PIN's for all entries in the user directory. Two new users joined your organization and you updated the directory with new users' information. For the new users to get certificates, the directory must contain PIN's. And you want to set PIN's for just those user entries without making changes to any of the other user entries. Instead of constructing a complex LDAP filter to filter out just these two entries, you can construct a general filter, put the two users' DN's in the input file, and run the PIN Generator.
- Assume that you want your users to use their social security numbers as PIN's. You can enter users' social security numbers as PIN's in the input file, and the PIN Generator will store them as hashed values in the directory.

The format of the input file is the same as that of the output file (see "Output File" on page 296), with the omission of the status line. In the input file, you can choose to specify PIN's for all the DN's in the file, for specific DN's, or for none of the DN's. If the PIN attribute is missing for a DN, the tool automatically generates a random PIN.

For example, you can set up your input file to look like this:

```
dn:cn=user1, o=netscape
<blank line>
dn:cn=user2, o=netscape
<blank line>
...
dn:cn=user3, o=netscape
```

You can also provide PINs, in plain-text format, for the DNs in the input file, which is then hashed according to the command-line arguments. For example, you can set up your input file to look like this:

```
dn:cn=user1, o=netscape
pin:pl229Ab
<blank line>
dn:cn=user2, o=netscape
pin:9j65dSf
<blank line>
...
dn:cn=user3, o=netscape
pin:3knAg60
<blank line>
```

**Note** You cannot provide hashed PINs to the tool.

## Output File

The PIN Generator can capture the output to a text file specified by the `output=<file_name>` argument.



The captured output will contain a sequence of records and will be in the following format:

```
dn: <user_dn>1
pin: <generated_pin>1
status: <status>1
<blank line>
dn: <user_dn>2
pin: <generated_pin>2
status: <status>2
<blank line>
...
dn: <user_dn>n
pin: <generated_pin>n
status: <status>n
<blank line>
```

where

`<user_dn>` is a distinguished name that matched the specified DN pattern (specified by the DN filter) or that was in the input file (the DN file). By default, the delimiter is ";" or the character defined on the command line.

`<generated_pin>` is a string of characters with either fixed or variable length, dependent on parameters passed into the command.

`<status>` is one of the values specified in Table 10.1 on page 294.

The first line in each record will always be the distinguished name. The subsequent lines, for `pin` and `status`, are optional. The record ends with a blank line. The end of line (EOL) sequence is as follows:

**Windows NT**    `\r\n`

**Unix**        `\n`

## How PINs Are Stored in the Directory

Each PIN is concatenated with the corresponding user's LDAP attribute named in the `saltattribute` argument. If this argument is not specified, the DN of the user is used. Then, this string is hashed with the hash routine specified in the `hash` argument (the default selection is SHA-1).

Then, one byte is prepended to indicate the hash type used. Here's how the PIN gets stored:

```
byte[0] = x
```

The value of `x` depends on the hash algorithm chosen during the PIN generation process:

`x=0` if the hash algorithm chosen is SHA-1.

`x=1` if the hash algorithm chosen is MD5.

`x=45` if the hash algorithm chosen is none.

```
byte[1...] = hash("DN"+"pin")
```

The PIN is stored in the directory as a binary value, not as a base-64 encoded value.

## Exit Codes

The PIN Generator returns exit codes to the shell window; for a list of codes, see Table 10.2. If you plan on automating the PIN-generation process, exit codes are useful in programming shell scripts.

**Table 10.2** Exit codes returned by the PIN Generator

Exit code	Description
0	Indicates that PIN generation was successful; that is, PINs are set for all the DN's in the specified directory.
2	Indicates that the tool could not open the certificate database specified by the <code>certdb</code> parameter.

Table 10.2 Exit codes returned by the PIN Generator (Continued)

Exit code	Description
3	Indicates that the tool could not locate the certificate specified by the <code>nickname</code> parameter in the specified certificate database.
4	Indicates that the tool could not bind to the directory as the user specified by the <code>binddn</code> parameter (over SSL).
5	Indicates that the tool could not open the output file specified by the <code>output</code> parameter.
7	Indicates an error parsing command-line arguments.
8	Indicates that the tool could not open the input file specified by the <code>input</code> parameter.
9	Indicates that the tool encountered an internal error.
10	Indicates that the tool found a duplicate entry in the input file specified by the <code>input</code> parameter.
11	Indicates that the tool didn't find the <code>salt</code> attribute, specified by the <code>saltattribute</code> parameter, in the directory.

## Generating PINs

Generating PINs for end entities is a six-step process:

- Step 1. Check the Directory for User Entries
- Step 2. Update the Directory Schema
- Step 3. Prepare the Input File
- Step 4. Run the Command Without the Write Option

- Step 5. Check the Output File
- Step 6. Run the Command Again with the Write Option

## Step 1. Check the Directory for User Entries

Before you run the PIN Generator, make sure that the directory you intend to use for generating PINs has been populated with all end-entity entries that require PINs. It is also a good idea to confirm with that directory's administrator that all pending directory requests have been processed before the PIN Generator starts its operation.

## Step 2. Update the Directory Schema

By default, the PIN Generator modifies the `pin` attribute in the directory's user entry (see "Arguments" on page 286). Because this attribute is not part of the standard `organizationalPerson`, you need to add it—that is, create a new object class in your authentication directory's schema.

In general, you need to update the `slapd.user_at.conf` file to include the `pin` attribute and the `slapd.user_oc.conf` file to include the object class definition. The modified schema should look similar to this:

```
attribute pin bin

objectclass pinPerson

    superior organizationalPerson

    allows
        pin
```

Netscape Directory Server provides the appropriate user interface to update its schema. It is recommended that you update the schema using the user interface (instead of editing the schema files directly). Sections that follow explain how to update the Directory Server 3.x and 4.x schema using the user interface. For complete information about the schema, see the appropriate Directory Server documentation.

**Important** Once you have modified the schema, you must restart Directory Server.

When the PIN Generator adds the PIN to the user entry, it also adds the appropriate object class. By default, the object class name is `pinPerson`, and the name of the attribute is `pin`, but you can override this with the object class and attribute options.

## Updating Netscape Directory Server 3.x Schema

If you are using Netscape Directory Server version 3.x as your authentication directory, you can follow the instruction below to update the schema:

1. Open a web browser window.
2. Enter the URL to the Directory Server instance that you want to use for storing end-entity PINs.
3. Click Schema.
4. In the left frame, click the "Edit or View Attributes" link.

The Manage Attributes form appears.



5. In the Add New Attribute section, enter the information for the new attribute:

**Attribute Name.** Type `pin` in this field.

**Attribute OID (optional).** Entering information in this field is optional. So, you can leave it blank.

**Syntax.** Choose `binary` from the drop-down list.

6. Click Add New Attribute.
7. In the left frame, click the Create Objectclass link.

The Create Objectclass form appears; this form allows you to create a new object class to hold the new attribute you added.

**Netscape Directory Server - slapd-pk - Netscape**

File Edit View Go Communicator Help

**NETSCAPE® DIRECTORY SERVER 3.0**  
©1997 Netscape Communications Corporation. All Rights Reserved.

Server Preferences Access Control Server Status Database Management Replication Schema

**Schema**  
[Edit or View Attributes](#)  
[Create Objectclass](#)  
[Edit or View Objectclasses](#)

### Create ObjectClass

**ObjectClass Name:**

**Parent Objectclass:**

**ObjectClass OID: (optional)**

**Available Attributes**

- abstract
- accountUnlockTime
- aci
- administratorContactInfo
- adminUrl
- aliasedObjectName
- altServer
- associatedDomain
- associatedName
- attributeTypes
- audio
- authorCn
- authorityRevocationList;binary
- authorSn
- buildingName
- businessCategory

**Required Attributes**

- objectclass
- sn
- cn

**Allowed Attributes**

- aci
- description
- seealso
- telephonenumber
- userpassword
- destinationindicator
- facsimiletelephonenumber
- internationalisdnnumber
- l
- ou

Buttons: Add -->, <-- Remove, Add -->, <-- Remove, Apply, Admin

Document: Done

8. Enter information as appropriate:

**ObjectClass Name.** Type `pinPerson` in this field.

**Parent Objectclass.** Choose `organizationalPerson` from the drop-down list.

**ObjectClass OID (optional).** Entering information in this field is optional. So, you can leave it blank.

9. In the Available Attributes list, select `pin` and then click the Add button shown to the left of the Allowed Attributes list.

This action moves the `pin` attribute to the Allowed Attributes list.

10. Click Create New ObjectClass.

The server informs you that the object class has been added to the schema.

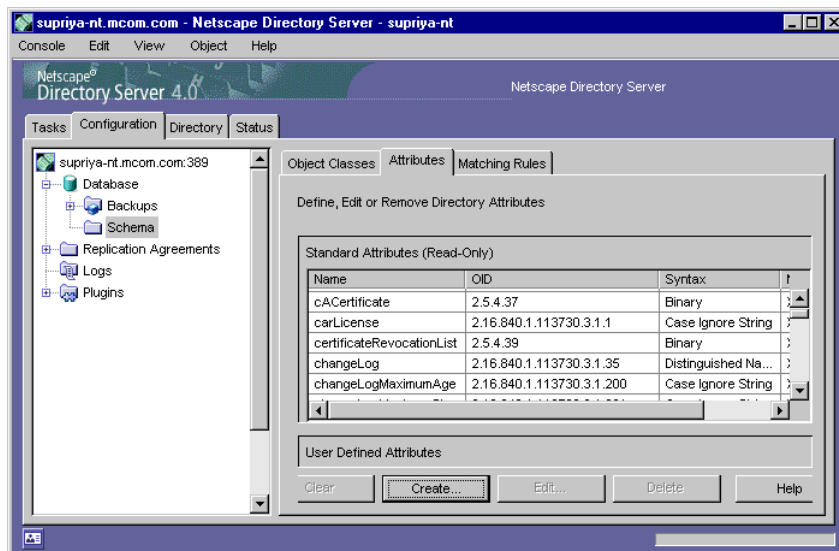
11. Restart the server, if prompted to do so.

## Updating Netscape Directory Server 4.x Schema

If you are using Netscape Directory Server version 4.x as your authentication directory, you can follow the instruction below to update the schema:

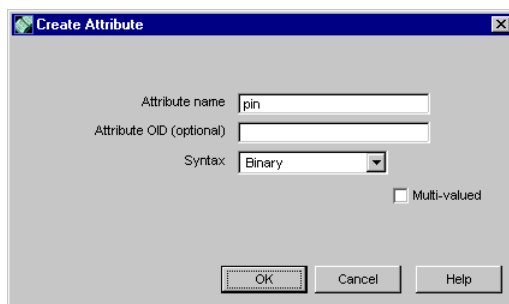
1. Access Netscape Console (see “Accessing Netscape Console” on page 53).
2. Select the directory that you want to use for storing end-entity PINs, and click Open to open the administration interface for Directory Server.
3. Click the Configuration tab.
4. In the navigation tree, expand the Database object.

- Click Schema, and then in the right pane, click the Attributes tab.



- Click Create.

The Create Attribute window appears.



- Enter information as appropriate:

**Attribute name.** Type `pin` in this field.

**Attribute OID (optional).** Entering information in this field is optional. So, you can leave it blank.

**Syntax.** Choose `binary` from the drop-down list.



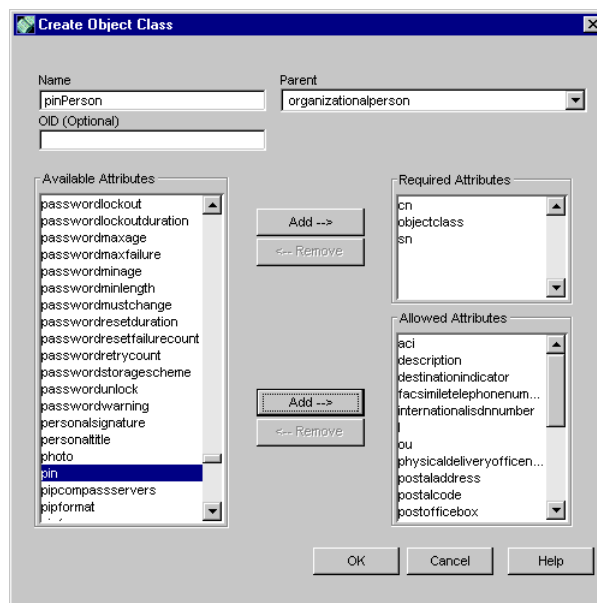
8. Click OK.

You are returned to the Attributes tab.

9. Click the Object Classes tab.

10. Click Create.

The Create Object Class window appears.



11. Enter information as appropriate:

**Name.** Type `pinPerson` in this field.

**Parent.** Choose `organizationalperson` from the drop-down list.

12. In the Available Attributes list, select `pin` and then click the Add button shown to the left of the Allowed Attributes list.

This action moves the `pin` attribute to the Allowed Attributes list.

13. Click OK.

The server informs you that the object class has been added to the schema.

14. Restart the server, if prompted to do so.

## Step 3. Prepare the Input File

This step is optional.

If you want to generate PINs for specific user entries or want to provide your own PINs, use an input file (to provide the tool with such information). For information on constructing an input file, see “Input File” on page 295.

## Step 4. Run the Command Without the Write Option

Run the `setpin` command without the `write` option. Be sure to include the `output` option. For details, see “The `setpin` Command” on page 286.

The tool will write PINs to the specified output file; no changes are made to the directory. This will give you the opportunity to check the PINs (by looking at the output file) before updating the directory.

To run the command:

1. Open a terminal window.
2. Go to `<server_root>/bin/cert/tools`  
  
    `<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.
3. Run the `setpin` command with the appropriate arguments.

## Step 5. Check the Output File

Check the output file to be sure it contains PINs for your users; the output should look similar to the one specified in “Output File” on page 296.

Verify that the tool has assigned PINs to the correct users and that the PINs conform to the length and character-set restrictions you specified. If the output isn't what you want, run the command again with appropriate arguments. Repeat the process until the output file shows the results you want.

## Step 6. Run the Command Again with the Write Option

When you are sure about the results, run the command again (with exactly the same arguments) with the `write` option and the `output` option. The tool stores the hashed PINs in the directory. For information on how PINs are stored in the directory, see “How PINs Are Stored in the Directory”.

Use the output file for delivering PINs to users; see “Delivering PINs to End Entities” on page 307.

## Delivering PINs to End Entities

After you have stored the PINs in the directory, you must deliver them to the users. To protect the privacy of PINs, be sure to use a secure, out-of-band mechanism for delivery. Here are a few suggested delivery mechanisms:

- Encrypted email (S/MIME)—if your company has S/MIME mail set up, you can deliver PINs to users by encrypted mail.
- Mail—you can mail PINs to users, for example along with their pay slips.
- Personal delivery—you can arrange a secure means of delivering the password to the user, or ask the user to collect it from you in person.



# Configuring Authentication for End Entities

Netscape Certificate Management System (CMS) provides a customizable authentication subsystem that supports various mechanisms for authenticating end entities. This chapter explains how to configure Certificate Management System to use specific authentication plug-in modules for authenticating end entities during certificate enrollment. The chapter also shows how end-entity authentication plug-in implementations and configured instances appear in the configuration file.

Before reading this chapter, you should have read the chapter “Introduction to Authentication” on page 259. In particular, you should be familiar with the various plug-in modules for authenticating end entities that come with Certificate Management System. If you are not, see “End-Entity Authentication During Certificate Enrollment” on page 265.

This chapter has the following sections:

- Authentication Management (page 310)
- Managing Authentication Instances (page 316)
- Managing Authentication Plug-in Modules (page 325)

# Authentication Management

You can manage end-entity authentication in two ways:

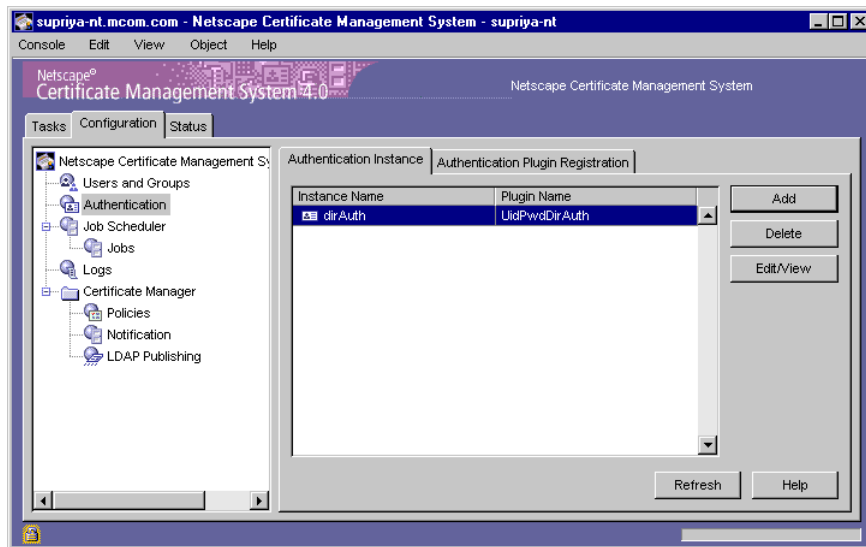
- By making changes to the authentication-specific parameters from the CMS window, explained in “Authentication Management from the CMS Window” on page 310
- By editing the parameters in the configuration file, explained in “Authentication Parameters in the Configuration File” on page 313

The recommended method is to use the CMS window.

## Authentication Management from the CMS Window

Figure 11.1 shows the CMS window, which provides the required user interface to support authentication management for end entities.

Figure 11.1 Authentication information in the CMS window



In the CMS window you will find a single Authentication object. This object represents the authentication plug-in implementations and instances (for end entities) currently recognized by this instance of Certificate Management System. From this window you can accomplish the following operations:

- Configuration of currently registered authentication plug-in modules
- Registration of new (custom) authentication plug-in modules

The sections that follow describe the parts of the window from which you carry out these operations.

## Authentication Instance Tab

The Authentication Instance tab lists the currently configured authentication instances, so that you can manage them at a single place. From this tab you can perform the following operations:

**Add.** The add operation shows a list of registered authentication plug-in modules from which you can select the one you want to configure. You can configure the selected module with the help of the authentication instance editor (see “Authentication Instance Editor” on page 312). When you save the changes, Certificate Management System creates the new authentication instance and displays it in the list of authentication instances. For instructions on adding new authentication instances, see “Adding an Authentication Instance” on page 317.

**Delete.** The delete operation allows you to remove unwanted authentication instances from the CMS configuration. For instructions on deleting authentication instances, see “Deleting an Authentication Instance” on page 322.

**Edit/View.** The edit operation allows you to view and modify the configuration parameter values of currently configured authentication instances. You modify the parameter values with the help of the authentication instance editor (see “Authentication Instance Editor” on page 312). For instructions on modifying authentication instances, see “Modifying an Authentication Instance” on page 322.

## Authentication Instance Editor

The authentication instance editor is designed to be generic. Its simple graphical interface enables you to create new instances and modify the configuration of an individual authentication instance. When you are *adding* a new instance, the editor shows the configuration parameters pertaining to the plug-in module you selected. When you are *modifying* an instance, the editor shows the configuration parameters pertaining to the instance you selected.

All configurable parameters are displayed in the form of a table with two columns and multiple rows, each parameter occupying a row in the table. The left column lists the names of the configurable parameters; the right column is designated for entering the appropriate values. The ordering of the configurable parameters is irrelevant unless it is defined by the authentication plug-in implementation.

The authentication instance editor provides normal save, cancel, and help functionality. You can specify *names* for authentication instances, but only at the time of adding new ones; you cannot change names later.

## Authentication Plugin Registration Tab

The Authentication Plugin Registration tab lists the currently registered authentication plug-in implementations for the selected CMS instance and gives you access to the window from which you can register new authentication plug-in modules. On this tab you will find the names of registered plug-in modules listed on the left and the path to the Java class that implements the plug-in module listed on the right.

You can perform the following operations from this tab:

**Register.** This operation allows you to register a new authentication plug-in module. You do this with the help of the authentication registration editor (see “Authentication Plug-in Registration Editor” on page 313).

When you save the changes, Certificate Management System loads the authentication plug-in implementation and displays it in the list of registered plug-ins. For instructions on registering new authentication plug-in modules, see “Registering an Authentication Plug-in Module” on page 325.



**Delete.** This operation allows you to remove unwanted authentication plug-in modules from the CMS framework. For instructions on deleting authentication plug-in modules, see “Deleting an Authentication Plug-in Module” on page 327.

## Authentication Plug-in Registration Editor

The authentication plug-in registration editor allows you to register new authentication plug-in modules in the CMS authentication framework. Registering a new authentication plug-in module involves specifying the name of the plug-in module and the full name of the Java class that implements the authentication interface (implementation must be on the class path).

For example, you can add an authentication implementation named as follows:

```
com.netscape.authentication.ssnAuth
```

## Authentication Parameters in the Configuration File

The sample shown in Figure 11.2 illustrates how authentication-specific information appears in the configuration file. Keep the following points in mind:

- All authentication-specific information, such as registered authentication implementations and configured instances, appears in the Authentication section of the configuration file. See the example shown in Figure 11.2 below.
- Each registered authentication plug-in module is identified by its implementation name.
- Each configured instance of an authentication implementation is identified by the name you specified when creating it.
- You can create multiple instances out of an implementation; each instance must have a unique name. For details, see “Authentication Plug-in Implementation and Instance” on page 314.

- The name of the authentication instance must be used in enrollment forms to identify the authentication manager. For details, see “Adding an Authentication Instance” on page 317.

Figure 11.2 End-entity authentication-specific information in the CMS configuration file

```
## Authentication Manager implementations for EE authentication

#Plug-in implementation for user ID and password-based authentication
auths.impl.UidPwdDirAuthentication.class=com.netscape.certsrv.authentication.UidPwdDirAuthentication

#Plug-in implementation for user ID, password, and PIN-based authentication
auths.impl.UidPwdPinDirAuthentication.class=com.netscape.certsrv.authentication.UidPwdPinDirAuthentication

## Authentication Manager instances for EE authentication

# Authentication manager for EEs in a directory named auth_dir.foo.com
auths.instance.ee_dir_auth_mgr1.implName=UidPwdDirAuthentication
auths.instance.ee_dir_auth_mgr1.Idap.dnPattern=E=$attr.mail.1, CN=$attr.cn, OU=$dn.ou.2, O=$dn.o, C=US
auths.instance.ee_dir_auth_mgr1.Idap.IdapAttributes=mail, mailalternateaddress
auths.instance.ee_dir_auth_mgr1.Idap.Idapconn.host=auth_dir.foo.com
auths.instance.ee_dir_auth_mgr1.Idap.Idapconn.port=389
auths.instance.ee_dir_auth_mgr1.Idap.Idapconn.secureConn=true
auths.instance.ee_dir_auth_mgr1.Idap.Idapconn.version=3
auths.instance.ee_dir_auth_mgr1.Idap.basedn=o=airius.com
auths.instance.ee_dir_auth_mgr1.Idap.minConns=2
auths.instance.ee_dir_auth_mgr1.Idap.maxConns=8
```

To change the configuration by editing the configuration file, follow the instructions in “Changing the Configuration by Editing the Configuration File” on page 72.

## Authentication Plug-in Implementation and Instance

Authentication managers are implemented as Java classes, which are then registered with Certificate Management System as plug-ins. You can use a given implementation of an end-entity authentication plug-in module and configure multiple instances of it. Each instance must have a unique name (an alphanumeric string with no spaces) and can contain different input parameter values to apply to different end-entity enrollment requests. In other words, a

given end-entity authentication implementation can be shared by multiple configurations. You can also distinguish the applicability of configured instances by including appropriate instance names.

For example, you can configure the `UidPwdDirAuth` plug-in module so that it authenticates users in two different directories. The figures that follow illustrate this capability. The plug-in named `UidPwdDirAuthentication` has been used to create two authentication instances, *ee\_dir\_auth\_mgr1* and *ee\_dir\_auth\_mgr2*, each of which authenticates end entities in a specific directory.

Figure 11.3 shows the two instances, both based on the same plug-in module, in the CMS window.

Figure 11.3 Authentication instances for two directories (CMS window)

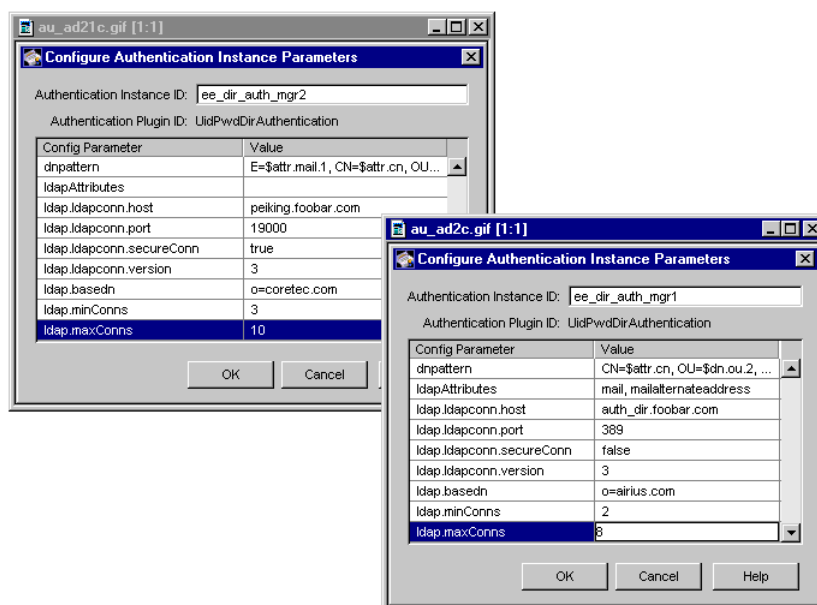


Figure 11.4 shows the two authentication instances, both based on the same plug-in module, in the configuration file.

Figure 11.4 Authentication instances for two directories (configuration file)

```
## Authentication manager for EEs in a directory named auth_dir.foobar.com

auths.instance.ee_dir_auth_mgr1.implName=UidPwdDirAuthentication
auths.instance.ee_dir_auth_mgr1.Idap.dnpattern=E=$attr.mail.1,CN=$attr.cn,OU=$dn.ou.2,O=$dn.o,C=US
auths.instance.ee_dir_auth_mgr1.Idap.IdapAttributes=mail,mailalternateaddress
auths.instance.ee_dir_auth_mgr1.Idap.Idapconn.host=auth_dir.foobar.com
auths.instance.ee_dir_auth_mgr1.Idap.Idapconn.port=389
auths.instance.ee_dir_auth_mgr1.Idap.Idapconn.secureConn=true
auths.instance.ee_dir_auth_mgr1.Idap.Idapconn.version=3
auths.instance.ee_dir_auth_mgr1.Idap.basedn=o=airius.com
auths.instance.ee_dir_auth_mgr1.Idap.minConns=2
auths.instance.ee_dir_auth_mgr1.Idap.maxConns=8

## Authentication manager for EEs in a directory named peiking.foobar.com

auths.instance.ee_dir_auth_mgr1.implName=UidPwdDirAuthentication
auths.instance.ee_dir_auth_mgr1.Idap.dnpattern=E=$attr.mail.1,CN=$attr.cn,OU=$dn.ou.1,O=$dn.o,C=US
auths.instance.ee_dir_auth_mgr1.Idap.IdapAttributes=
auths.instance.ee_dir_auth_mgr1.Idap.Idapconn.host=peiking.foobar.com
auths.instance.ee_dir_auth_mgr1.Idap.Idapconn.port=19000
auths.instance.ee_dir_auth_mgr1.Idap.Idapconn.secureConn=false
auths.instance.ee_dir_auth_mgr1.Idap.Idapconn.version=3
auths.instance.ee_dir_auth_mgr1.Idap.basedn=o=coretec.com
auths.instance.ee_dir_auth_mgr1.Idap.minConns=3
auths.instance.ee_dir_auth_mgr1.Idap.maxConns=10
```

## Managing Authentication Instances

This section explains how to use the CMS window to perform the following operations:

- Adding an Authentication Instance
- Deleting an Authentication Instance
- Modifying an Authentication Instance

For information on adding or changing authentication-specific information in the configuration file, see “Authentication Parameters in the Configuration File” on page 313.

## Adding an Authentication Instance

Adding an authentication instance to the CMS configuration involves creating a new instance of an already registered plug-in module, assigning a unique name (an alphanumeric string with no spaces) to the instance, and entering appropriate values for the parameters that define the plug-in implementation you want to create an instance of.

When you add an authentication instance, the CMS configuration is updated with authentication-specific information only. The server does not associate the authentication instance you added with any of the end-entity enrollment forms; that is, the end-entity servlets that should use this authentication instance are not configured yet. For the new authentication instance to work with end-entity enrollment forms, you must update the appropriate forms, as follows:

- If you are using custom forms for end-entity enrollment, be sure to include the following line in the forms, and replace `myAuthMgr` with the name of the authentication instance you added.

```
<INPUT TYPE="HIDDEN" NAME="authenticator" VALUE="myAuthMgr">
```

For example, if your authentication instance is named `testAuth`, the line would look like this:

```
<INPUT TYPE="HIDDEN" NAME="authenticator" VALUE="testAuth">
```

- If your end-entity forms are based on the HTML forms that come with Certificate Management System, locate the above-mentioned line; it is already embedded in the forms. Then replace `myAuthMgr` with the name of the authentication instance you added.

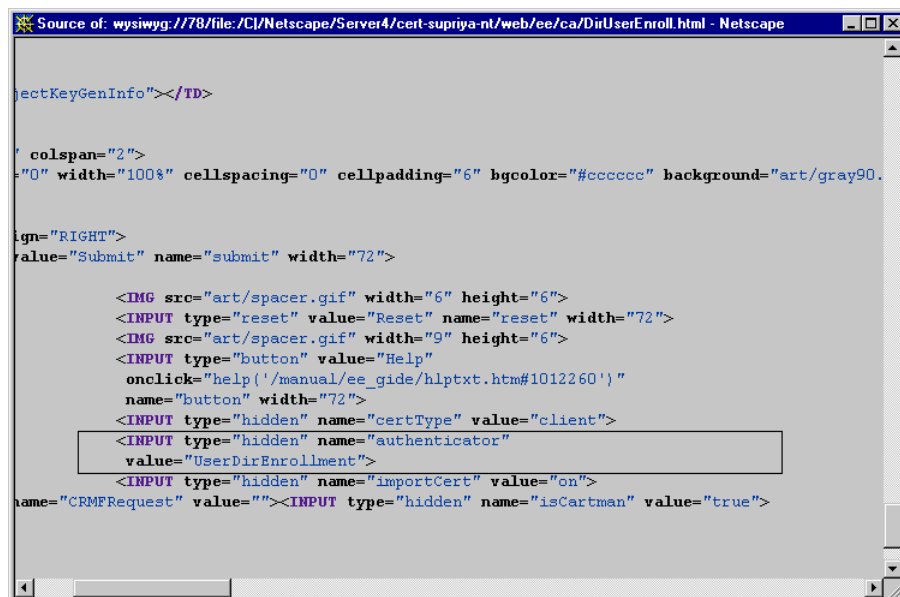
By default, the enrollment forms include the authentication instance names listed in Table 11.1. Note that the authentication instances are not created by default; only the instance names are embedded in the forms. If you create authentication instances with these names, you don't have to update the enrollment forms to replace `myAuthMgr` with the name of the authentication instance.

Table 11.1 Default authentication instance names embedded in enrollment forms

Enrollment form (filename)	Authentication instance name
Directory-based enrollment for end users (DirUserEnroll.html)	UserDirEnrollment
Directory- and PIN-based enrollment for end users (DirPinUserEnroll.html)	PinDirEnrollment
Directory-based enrollment for servers (DirServerEnroll.html)	serverDirEnrollment

Figure 11.5 shows the default directory-based enrollment form configured to use an authentication instance identified as UserDirEnrollment.

Figure 11.5 Authentication information in the default directory-based enrollment form



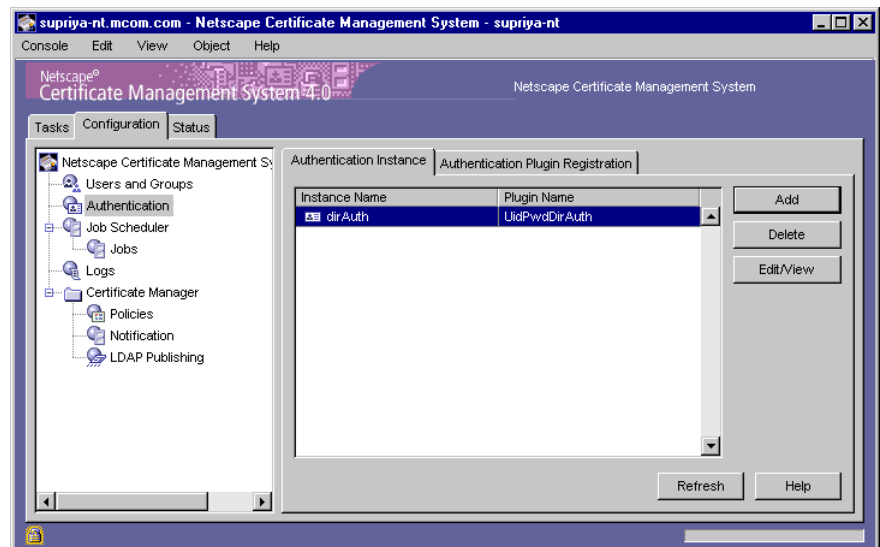
For information on locating and customizing the default end-entity forms, see “Summary of End-Entity Forms and Templates” on page 553.

**Note** If you do not configure Certificate Management System to use any of the authentication plug-in modules listed in the Authentication Plugin Registration tab, the server uses manual authentication for end-entity enrollment. This means that all end-entity enrollment requests are queued for agent approval. For more information, see “Manual Authentication” on page 266.

To add an authentication instance to the CMS configuration:

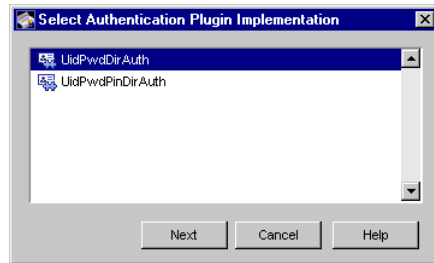
1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, click Authentication.

The right pane shows the Authentication Instance tab, which lists any currently configured authentication instances. For information about this tab, see “Authentication Instance Tab” on page 311.



4. Click Add.

The Select Authentication Plugin Implementation window appears. It lists the currently registered authentication plug-in modules.



5. Select a plug-in module.

The following choices are the ones provided out of the box with Certificate Management System (they are described in “End-Entity Authentication During Certificate Enrollment” on page 265). If you have registered any custom authentication plug-in modules, they too will be available for selection.

- UidPwdDirAuth

This entry represents the user ID- and password-based authentication module explained in “Directory-Based Authentication” on page 268.

- UidPwdPinDirAuth

This entry represents the user ID-, password-, and PIN-based authentication module (with or without PIN removal) explained in “Directory-Based Authentication with PINs” on page 275.

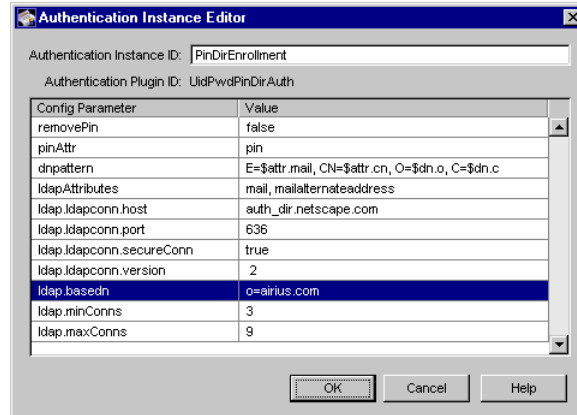
To configure Certificate Management System for PIN-based authentication, you must have an LDAP-compliant directory deployed (with end-entity entries in it) and a unique PIN for each end entity.

For the purposes of this instruction, assume that you selected UidPwdPinDirAuth.



6. Click Next.

The Configure Authentication Instance Parameters window appears. It lists the configuration information required for this authentication instance. For more information on how this window functions, see “Authentication Instance Editor” on page 312.



7. In the Authentication Instance ID field, type a unique name for this instance that will help you identify it.

For the name, be sure to use an alphanumeric string with no spaces.

8. In the configuration area, specify the required information by filling in parameter values in the corresponding text fields (the right column).

If you do not want to set any restrictions on a particular parameter, leave its value field blank.

9. Click OK.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

## Deleting an Authentication Instance

You can delete any unwanted authentication instances from the CMS configuration. If you delete an authentication instance, the associated end-entity enrollment forms, if used, fail to authenticate end entities. If you want these forms to work with other authentication instances, make the appropriate changes to the forms; see “Step 5. Customize the End-Entity Enrollment Forms” on page 340.

To delete an authentication instance from the CMS configuration:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, click Authentication.

The right pane shows the Authentication Instance tab, which lists currently configured authentication instances. For information about this tab, see “Authentication Instance Tab” on page 311.

4. In the Instance Name list, select the instance you want to delete and click Delete.
5. When prompted, confirm the delete action.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

## Modifying an Authentication Instance

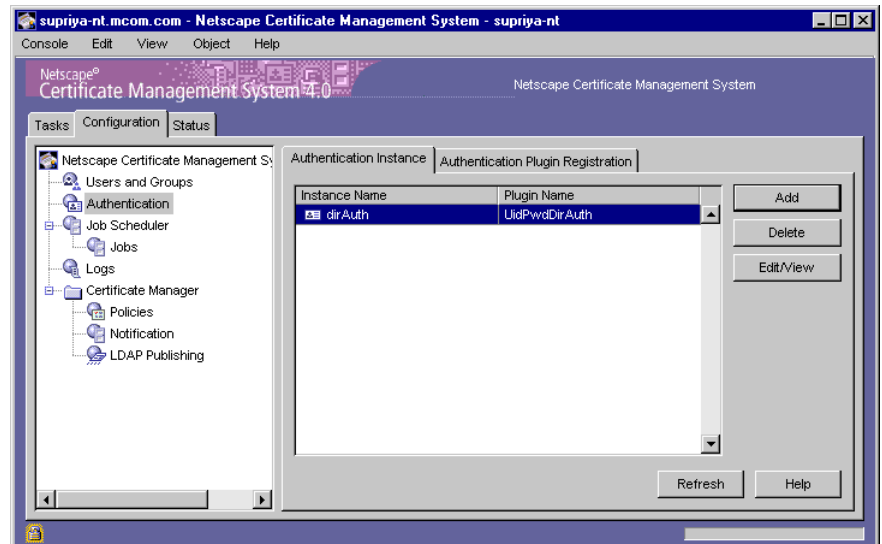
Modifying an authentication instance involves changing its configuration parameter values; you cannot change the name of an instance. To change the name of an instance, create a new instance using the same authentication plug-in module that you used to create the instance you want to rename, with the same parameter values, and delete the old one.

When you modify an authentication instance, the CMS configuration gets updated with authentication-specific information. Because you are not changing the name of the authentication instance, you do not have to make any changes to the end-entity servlet configuration.

To modify an authentication instance in the CMS configuration:

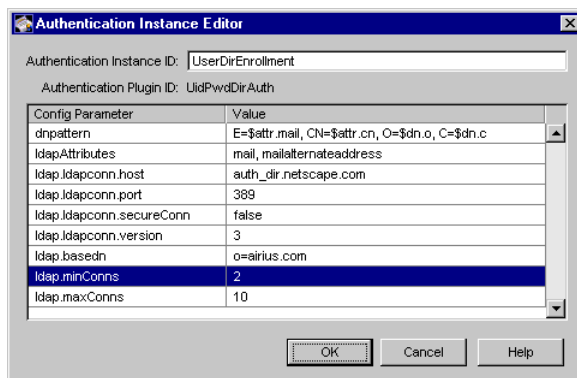
1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, click Authentication.

The right pane shows the Authentication Instance tab, which lists currently configured authentication instances.



4. In the Instance Name list, select the instance you want to modify and click Edit.

The Configure Authentication Instance Parameters window appears, showing the current configuration of this instance. For more information on how this window functions, see “Authentication Instance Editor” on page 312.



5. Make the necessary changes by filling in parameter values in the corresponding text fields (the right column).

If you do not want to set any restrictions on a particular parameter, leave its value field blank.

- For a description of configuration parameters pertaining to the instances created using the UidPwdDirAuth plug-in module, see Table 9.1 on page 272.
- For a description of configuration parameters pertaining to the instances created using the UidPwdPinDirAuth plug-in module, see Table 9.2 on page 277.

6. Click OK.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

# Managing Authentication Plug-in Modules

This section explains how to use the CMS window to perform the following operations:

- Registering an Authentication Plug-in Module
- Deleting an Authentication Plug-in Module

For information on adding or changing authentication-specific information in the configuration file, see “Authentication Parameters in the Configuration File” on page 313.

## Registering an Authentication Plug-in Module

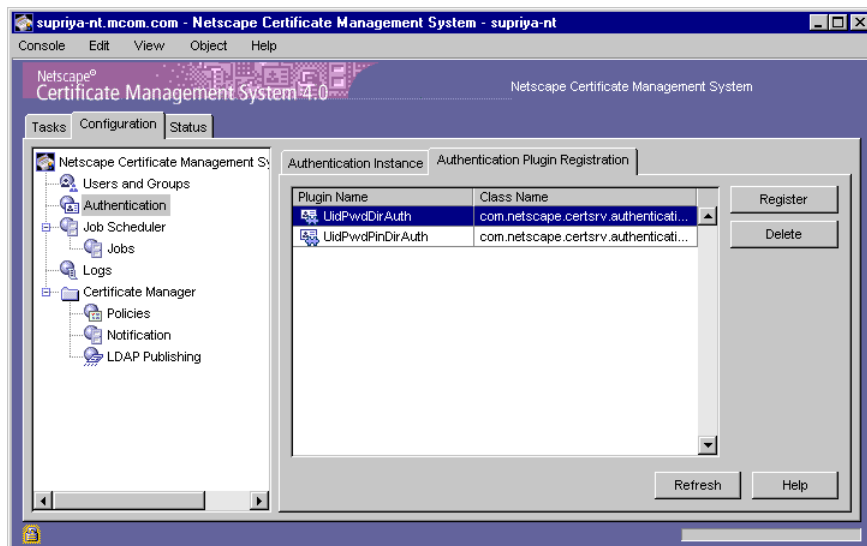
You can register custom authentication plug-in modules from the CMS window. Before registering an authentication plug-in, be sure to put the Java class for the plug-in module in the `classes` directory; see “Compiling and Installing Authentication Manager Plug-ins” on page 335.

To register an authentication plug-in module in the CMS authentication framework:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.

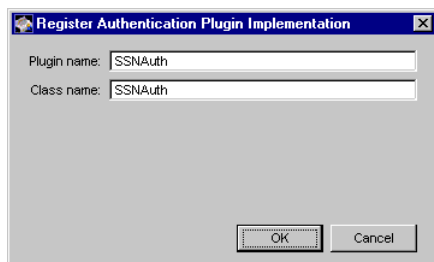
3. In the navigation tree, click Authentication, and in the right pane, click the Authentication Plugin Registration tab.

The Authentication Plugin Registration tab lists currently registered plug-in modules. For information about this tab, see “Authentication Plugin Registration Tab” on page 312.



4. Click Register.

The Register Authentication Plugin Implementation window appears. For information on how this window works, see “Authentication Plug-in Registration Editor” on page 313.



5. Specify the appropriate information:

**Plugin name.** Type the name of the plug-in module.

**Class name.** Type the full name of the class for this plug-in module—that is, the path to the implementing Java class. If this class is part of a package, be sure to include the package name. For example, if you are registering a class named `NISAuth` and if this class is in a package named `com.mycompany`, type `com.mycompany.NISAuth`.

6. Click OK.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

## Deleting an Authentication Plug-in Module

You can delete unwanted authentication plug-in modules by using the CMS window. Before deleting a plug-in module, be sure to delete all the instances that are based on this plug-in; see “Deleting an Authentication Instance” on page 322. You should also update the appropriate end-entity enrollment forms.

To delete an authentication plug-in module from the CMS authentication framework:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, click Authentication, and in the right pane, click the Authentication Plugin Registration tab.

The Authentication Plugin Registration tab lists currently registered plug-in modules. For information about this tab, see “Authentication Plugin Registration Tab” on page 312.

4. In the Plugin Name list, select the plug-in module you want to delete and click Delete.

5. When prompted, confirm the delete action.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.



# Developing Authentication Plug-ins

The authentication subsystem evaluates and authenticates the identity of an end entity that is requesting service from Netscape Certificate Management System (CMS). By default, the authentication subsystem supports various authentication mechanisms; these are explained in “Introduction to Authentication” on page 259.

If the authentication plug-in modules provided with Certificate Management System do not meet your requirements, you can write custom authentication plug-ins and plug them into the authentication framework of Certificate Management System.

This chapter provides an overview of the authentication subsystem architecture, discusses the interfaces for writing custom authentication plug-ins (called *authentication managers*), and explains how to implement custom authentication.

The chapter has the following sections:

- Authentication Subsystem Architecture (page 330)
- Customizing Authentication (page 333)

**Note** To customize authentication, you must be familiar with programming in Java and the Netscape Certificate Management System 4.x software development kit (SDK).

# Authentication Subsystem Architecture

The authentication subsystem of Certificate Management System permits a flexible number of authentication managers for end-entity authentication. This architecture is illustrated in Figure 12.1. An authentication manager (authMgr) is a configured instance of an authentication plug-in implementation.

Figure 12.1 Authentication subsystem architecture

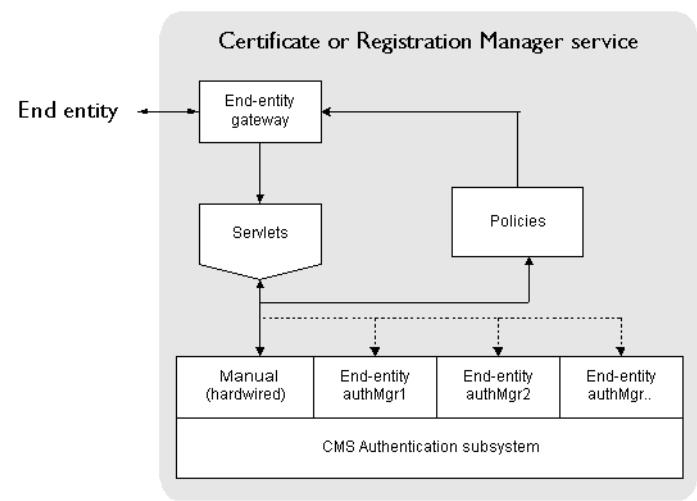


Table 12.1 describes the components shown in Figure 12.1.

Table 12.1 Description of components

Component	Description
End entity	Specifies end entities, such as users, servers, virtual private network (VPN) clients, or routers enrolling for a certificate through a web-based form.
End-entity gateway	Specifies the end-entity gateway provided by Certificate Management System.

Table 12.1 Description of components (Continued)

Component	Description
Servlets	Specifies end-entity servlets. Each type of request from an end entity is handled by a different servlet; each servlet serves the correct HTML form or other information to the end entity, depending on the protocols supported by that entity.
Policies	Specifies currently configured policies for a Certificate Manager or Registration Manager (depending on which subsystem is processing end-entity requests).
Manual	Specifies manual authentication. If you do not configure Certificate Management System to use any of the authentication plug-in modules listed in the Authentication Registration Plugin tab, the server uses manual authentication for end-entity enrollment. This means that all end-entity enrollment requests are queued for agent approval.
End-entity authMgr	Specifies configured authentication managers (based on authentication plug-ins provided out of the box and custom, if any) for authenticating end entities.

## How the Architecture Works

The setup shown in Figure 12.1 works in the following way:

1. An end entity enrolls for a certificate through a web-based form that includes fields in which the end entity enters the required enrollment credentials or provides authentication parameter values. The form has been customized to use a specific authentication manager.
2. On the CMS side, a servlet receives the following as input from the form the end entity used:
  - The name of the authentication manager
  - An attribute set (or authentication parameters)

The servlet then calls the appropriate authentication manager instance to authenticate the end entity.

- 3. After the end entity is authenticated, policies are applied on the request by either a Certificate Manager or Registration Manager, depending on which subsystem is processing the request.
- 4. Finally, a certificate is issued and returned to the end entity.

## How Authentication Managers Are Used

To authenticate an end entity, an enrollment servlet calls the authentication manager `authenticate (IAuthCredentials)` method, passing it credentials received in the HTTP input (the HTML form). The authentication manager implementation specifies which HTTP input variables are required for authentication. For example, in the directory-based authentication manager the required credentials are UID and password. If authentication succeeds, an `AuthToken` is returned. Otherwise, an exception is thrown. `IAuthCredentials` and `AuthToken` contain lists of attribute and value pairs.

The `AuthToken` returned contains the fields and values listed in Table 12.2. The `tokenCertSubject` value must have the subject DN of the certificate set. Values for all other fields are optional.

Table 12.2 `AuthToken` fields and values

Field	Object type returned
<code>tokenCertSubject</code>	<code>netscape.security.X509.X500Name</code> (a Java <code>X509Name</code> object)
<code>tokenCertNotBefore</code>	<code>java.util.Date</code> (a Java <code>Date</code> object)
<code>tokenCertNotAfter</code>	<code>java.util.Date</code> (a Java <code>Date</code> object)
<code>tokenCertExts</code>	<code>netscape.security.X509.Certificate</code> <code>extensions</code> (a Java <code>X509Extensions</code> object)

For certificate enrollment the `AuthToken` must contain a field containing a certificate subject name for the authenticated user, for requests made through the `KEYGEN` tag. (Certificate Management System supports certificate requests in `KEYGEN`, `PKCS #10`, `CRMF` and `CEP` formats). The token field is `AuthToken.TOKEN_CERT_SUBJECT`. The value must be an instance of `netscape.security.x509.X500Name`. For other request formats that contain a subject name, a certificate subject name returned in the `AuthToken` will override the subject name in the request, for example in a `PKCS #10` request.

In addition, all `AuthTokens` contain the following:

- `authMgrInstName`—the name of the authentication manager (the instance) that authenticated the request
- `authMgrImplName`—the authentication manager's plug-in name (the implementation)
- The time of authentication

Optionally, the `AuthToken` can contain other attributes that customized policies can use in building the certificate.

After authentication, the `AuthToken` and the attributes and values returned in the `AuthToken` are stored in the request. Credentials from the form are discarded. The request is then passed to policies and then to the Certificate Manager for issuance.

## Customizing Authentication

Customizing authentication is a five-step process:

- Step 1. Decide on an Authentication Scheme
- Step 2. Write the Authentication Plug-in Module
- Step 3. Register the Authentication Manager Plug-in Module
- Step 4. Create an Instance of the Authentication Plug-in Module
- Step 5. Customize the End-Entity Enrollment Forms

## **Step 1. Decide on an Authentication Scheme**

When you consider customizing authentication, the first thing to decide is the kind of authentication scheme that you want for your end entities when they enroll for a certificate. In planning the scheme, you must identify and decide on the end-entity attributes Certificate Management System should retrieve from an end-entity request and evaluate the values of. In other words, in this step you must fix the attributes for authentication.

For example, if you have a corporatewide user database, you may want the end entities to provide their user IDs and passwords for the directory as authentication credentials. In that case, you would make sure that the form the end entities use for certificate enrollment includes user ID and password fields. Then end entities will provide this information when they request a certificate from Certificate Management System. Alternatively, you may want to issue certificates to any individual who has a social security number or other kind of valid identification, such as a driver's license or bank account.

## **Step 2. Write the Authentication Plug-in Module**

Authentication managers are implemented as Java classes, which are then registered with Certificate Management System as plug-ins. After you decide on the attributes for authenticating end entities, you need to write an authentication plug-in (Java class) that uses those attributes. Keep in mind that the plug-in implementation must conform to the CMS interface, as explained in the section that follows.

### **Authentication Manager Plug-in API**

To enable you to write custom authentication plug-ins, Certificate Management System provides an authentication manager plug-in API and related classes (Java Docs).

Your authentication manager plug-in must be a Java class that implements the following interface:

```
com.netscape.certsrv.authentication.IAuthManager
```

For the definition of `IAuthManager`, check the directory named `SDK` on the product CD. You can also download the CMS SDK from this site:

<http://home.netscape.com/eng/server/cms>

## Compiling and Installing Authentication Manager Plug-ins

When you are compiling an authentication manager plug-in using `javac`, be sure to include CMS classes in the classpath. For example, if a CMS instance named `testCA` is installed in `C:\netscape\server4` (default server root in Windows NT) or `usr/netscape/server4` (default server root in Unix), use the following to compile the authentication manager plug-in implementation:

**Windows NT**

```
> set CERT40DIR=C:\netscape\server4\bin\cert\jars
> javac -classpath \
C:\jdk1.1.6\lib\classes.zip;%CERT40DIR%/
ldapjdk.jar;%CERT40DIR%/certsrv.jar \
myAuthMgr.java
```

**Unix**

```
$ set CERT40DIR=usr/netscape/server4/bin/cert/jars
$ javac -classpath \
/usr/jdk1.1.6/lib/classes.zip:$CERT40DIR/
ldapjdk.jar:$CERT40DIR/certsrv.jar \
myAuthMgr.java
```

After compiling an authentication manager plug-in, add it to the CMS authentication framework as explained in the section that follows.

## Adding the Authentication Manager Class File to the CMS Authentication Framework

There are two ways in which you can add a custom authentication module class file to the CMS authentication framework. You can either put the class file in the server's default classpath or edit the server's `start-cert` script to include the path to your authentication plug-in module. The recommended method is that you add your class file to the server's default classpath.

The classes directories are located here:

`<server_root>/bin/cert/classes/...` and

`<server_root>/cert-<instance_id>/classes/...`

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

`<instance_id>` is the ID for this instance of Certificate Management System. You first specified this when you installed this server.

- If your authentication plug-in is installation specific—which means, it will be used by all CMS instances under a specific server root—put the class file in this directory:

`<server_root>/bin/cert/classes/...`

- If your authentication manager is instance specific—which means, it will be used by a specific CMS instance—put the class file in this directory:

`<server_root>/cert-<instance_id>/classes/...`

Alternatively, for testing purposes, you can also edit the classpath in the `start-cert` script to point to the custom class file.

**Important** Keep in mind that the changes you make to the `start-cert` script will take effect only when starting the server from the command line.

The default classpath in the `start-cert` script is as follows:

**Windows NT** `C:\Netscape\Server4\cert-testCA\classes\;C:\Netscape\Server4\bin\cert\classes\;C:\Netscape\Server4\bin\cert\jars\jss.jar;C:\Netscape\Server4\bin\cert\jars\certsrv.jar;C:\Netscape\Server4\java\ldapjdk.jar;C:\Netscape\Server4\bin\base\jre\lib\rt.jar;C:\Netscape\Server4\bin\base\jre\lib\i18n.jar;C:\Netscape\Server4\bin\cert\jars\jssjdk12.jar;C:\Netscape\Server4\java\swingall.jar`

**Unix** `/usr/netcape/server4/cert-testCA/classes/:/usr/netcape/server4/bin/cert/classes/:/usr/netcape/server4/bin/cert/jars/jss.jar:/usr/netcape/server4/bin/cert/jars/certsrv.jar:/usr/netcape/server4/java/ldapjdk.jar:/usr/netcape/server4/bin/base/jre/lib/rt.jar:/usr/netcape/server4/bin/base/jre/lib/i18n.jar:/usr/netcape/server4/bin/cert/jars/jssjdk12.jar`



To add a classpath to the start-cert script in Unix:

1. Go the command-line window.
2. Go to the CMS-instance directory. For example, `/usr/netscape/server4/cert-testCA`

3. Enter the following line at the prompt:

```
cat start-cert
```

You should see something similar to this:

```
#!/bin/sh

/usr/netscape/server4/bin/cert/admin/bin/start -i testCA
-r /usr/netscape/server4 -e -classpath

/usr/netscape/server4/bin/cert/classes:/usr/netscape/
server4/bin/cert/jars/jss.jar:/usr/netscape/server4/bin/
cert/jars/certsrv.jar:/usr/netscape/server4/java/
ldapjdk.jar:/usr/netscape/server4/bin/base/jre/lib/
rt.jar:/usr/netscape/server4/bin/base/jre/lib/i18n.jar:/
usr/netscape/server4/bin/cert/jars/jssjdk12.jar
```

4. Add your class's directory path to the start-cert script. Be sure to add the directory path to the beginning as shown in the example.

```
#!/bin/sh

/usr/netscape/server4/bin/cert/admin/bin/start -i testCA
-r /usr/netscape/server4 -e -classpath

<your_class's_directory_path>:/usr/netscape/server4/cert-
testCA/classes:/usr/netscape/server4/bin/cert/classes:/
usr/netscape/server4/bin/cert/jars/jss.jar:/usr/netscape/
server4/bin/cert/jars/certsrv.jar:/usr/netscape/server4/
```

```
java/ldapjdk.jar:/usr/netscape/server4/bin/base/jre/lib/
rt.jar:/usr/netscape/server4/bin/base/jre/lib/i18n.jar:/
usr/netscape/server4/bin/cert/jars/jssjdk12.jar
```

For example, if your class file is in a directory /u/jdoe/myAuthMgrs, here's how the start-cert script would look:

```
#!/bin/sh

/usr/netscape/server4/bin/cert/admin/bin/start -i testCA
-r /usr/netscape/server4 -e -classpath

/u/jdoe/myAuthMgrs:/usr/netscape/server4/cert-testCA/
classes:/usr/netscape/server4/bin/cert/classes:/usr/
netscape/server4/bin/cert/jars/jss.jar:/usr/netscape/
server4/bin/cert/jars/certsrv.jar:/usr/netscape/server4/
java/ldapjdk.jar:/usr/netscape/server4/bin/base/jre/lib/
rt.jar:/usr/netscape/server4/bin/base/jre/lib/i18n.jar:/
usr/netscape/server4/bin/cert/jars/jssjdk12.jar
```

To add a classpath to the start-cert.bat script in Windows NT:

1. Go to the command-line window.
2. Go to the CMS instance directory. For example,  
C:\netscape\server4\cert-testCA
3. Enter the following line at the prompt:

```
type start-cert.bat
```

You should see something similar to this:

```
net start cert-testCA /cC:\Netscape\Server4\cert-
testCA\classes\;C:\Netscape\Server4\bin\cert\classes\;C:\
Netscape\Server4\bin\cert\jars\jss.jar;C:\Netscape\Server
4\bin\cert\jars\certsrv.jar;C:\Netscape\Server4\java\ldap
jdk.jar;C:\Netscape\Server4\bin\base\jre\lib\rt.jar;C:\Ne
tscape\Server4\bin\base\jre\lib\i18n.jar;C:\Netscape\Serv
er4\bin\cert\jars\jssjdk12.jar;C:\Netscape\Server4\java\
swingall.jar
```

4. Add your class's directory path to the start-cert command. Be sure to add the directory path to the beginning as shown in the example.

```
net start cert-testCA /
c<your_class's_directory_path>C:\Netscape\Server4\cert-
testCA\classes\;C:\Netscape\Server4\bin\cert\classes\;C:\
Netscape\Server4\bin\cert\jars\jss.jar;C:\Netscape\Server
```

```
4\bin\cert\jars\certsrv.jar;C:\Netscape\Server4\java\ldap
jdk.jar;C:\Netscape\Server4\bin\base\jre\lib\rt.jar;C:\Ne
tscape\Server4\bin\base\jre\lib\i18n.jar;C:\Netscape\Serv
er4\bin\cert\jars\jssjdk12.jar;C:\Netscape\Server4\java\
swingall.jar
```

For example, if your class file is in a directory

C:\jdoe\myAuthMgrs\... here's how the start-cert.bat script would look:

```
net start cert-testCA /
cC:\jdoe\myAuthMgrs\;C:\Netscape\Server4\cert-
testCA\classes\;C:\Netscape\Server4\bin\cert\classes\;C:\
Netscape\Server4\bin\cert\jars\jss.jar;C:\Netscape\Server
4\bin\cert\jars\certsrv.jar;C:\Netscape\Server4\java\ldap
jdk.jar;C:\Netscape\Server4\bin\base\jre\lib\rt.jar;C:\Ne
tscape\Server4\bin\base\jre\lib\i18n.jar;C:\Netscape\Serv
er4\bin\cert\jars\jssjdk12.jar;C:\Netscape\Server4\java\s
wingall.jar
```

## Authentication Manager Examples

To aid you in writing custom authentication managers, Certificate Management System provides sample authentication plug-ins. You can find them in this directory:

```
<server_root>/bin/cert/samples/authentication/...
```

<server\_root> is the directory where the CMS binaries are kept. You first specified this directory during installation.

You can also download the samples from this site:

```
http://home.netscape.com/eng/server/cms
```

The samples directory includes an authentication manager that checks the user's user ID and password in a directory configured for authenticating users.

## Step 3. Register the Authentication Manager Plug-in Module

You can register custom authentication manager plug-in modules by using the CMS window. Before registering a custom plug-in, be sure to put the Java class for the plug-in in the `classes` directory. For instructions, see “Registering an Authentication Plug-in Module” on page 325.

## Step 4. Create an Instance of the Authentication Plug-in Module

After you have registered the custom authentication plug-in module, you must configure an instance of it. For instructions, see “Adding an Authentication Instance” on page 317.

## Step 5. Customize the End-Entity Enrollment Forms

After you have created the authentication manager instance, you must customize the appropriate HTML form for end-entity enrollment. Make the following changes:

- Update the enrollment forms to include all the required attributes.
- Update the enrollment forms to use the new authentication manager instance. This involves setting the value of `authenticator`, an HTML element in the enrollment form, to the name of the custom authentication manager instance.

The most convenient way to do this is by editing the forms provided out of the box for end entities. For example, you can add new fields to the directory-based authentication form provided for end-entity enrollment. You can also use custom enrollment forms, but if you do so, be sure to include the following HTML element in your end-entity enrollment form:

```
<INPUT TYPE="HIDDEN" NAME="authenticator" VALUE="myAuthMgr">
```

where myAuthMgr is the name of the authentication instance you want to use with the enrollment form.

Figure 12.2 shows a directory-based authentication form customized to use a social security number (SSN) for authentication in addition to user ID and password for the authentication directory. Note the text field named SSN.

Figure 12.2 End-entity enrollment form for SSN and directory-based authentication

**Directory And SSN Based Certificate Enrollment Form - Netscape**

File Edit View Go Communicator Help

---

**Directory And Pin Based Certificate Enrollment**

This form will help you put together the information that you need for requesting a personal certificate. Your directory credentials (i.e login and password) and the one time password your administrator communicated to you are required for automatic certificate issuance.

**Important:** When making this request you must use the Navigator/Communicator in which you wish to use the certificate.

**User's identity**

Please enter your user id and password for the corporate directory. This information will be used to verify your identity and privileges and to fetch information from the directory for constructing the subject name to include in the certificate.

User id

Password

Please enter your social security number.

SSN

**Certificate usage**

Please select one or more usage types for the certificate that you are requesting.

☐ Secure email

☒ SSL client

☐ Object signing

**User's public key information**

This form will cause your navigator to generate a private key and public key pair when you submit it. Your navigator retains the private key. The public component is submitted for certification. Please select the length of the key to generate below

Document: Done



# 5

## *Job Scheduling and Notification*

*Chapter 13* Introduction to Job Scheduling and Notifications

*Chapter 14* Configuring Jobs





# Introduction to Job Scheduling and Notifications

Netscape Certificate Management System (CMS) includes a component called *Job Scheduler* that can execute specific jobs at specified times. The job scheduler functions similar to a traditional Unix *cron* daemon in that it takes registered cron jobs and executes them at a preconfigured date and time. If configured, the scheduler checks at specified intervals for jobs waiting to be executed; if the specified execution time has arrived, the scheduler initiates the job automatically. Jobs that you might want to schedule include email notifications of timed events (such as the expiration of a certificate) that require action on the part of users, and periodic activities such as updates of related directories.

You can also configure Certificate Management System to send email notifications automatically to end entities, agents, or administrators when certain events occur. Unlike jobs that are executed on a preconfigured schedule, these notifications are event-driven—that is, whenever an event occurs, the server notifies the user. Notifiable events include certificate issuance and pending requests in an agent queue.

This chapter describes the job plug-in modules and event notifications that come with Certificate Management System and explains how to schedule times for jobs.

The chapter has the following sections:

- Built-in Job Plug-in Modules (page 346)
- Event-Driven Notifications (page 364)
- Customizing Notification Messages (page 370)

# Built-in Job Plug-in Modules

Certificate Management System comes with various job plug-in modules that can be employed by the server to automate certain activities. Table 13.1 lists these modules and indicates which subsystems can use them.

Table 13.1 Schedulable job plug-in modules for Certificate Manager and Registration Manager

Plug-in module name	Description
RenewalNotificationJob	<p>A schedulable job that notifies end entities by email that their certificates are about to expire and must be renewed, and optionally sends a summary of these notices to agents. For more information, see “Certificate Renewal Notifications” on page 347.</p> <p>Both the Certificate Manager and Registration Manager provide this plug-in module.</p>
RequestInQueueJob	<p>A schedulable job that notifies agents at regular intervals of the current state of the request queue.</p> <p>In addition, agents can also be notified by email that a request has been added to the request queue by configuring an event-driven notification. See “Notification of Request Queue Status” on page 353.</p> <p>Both the Certificate Manager and Registration Manager provide this plug-in module.</p>

Table 13.1 Schedulable job plug-in modules for Certificate Manager and Registration Manager (Continued)

Plug-in module name	Description
UnpublishExpiredJob	A schedulable job that updates the configured publishing directory periodically by removing expired certificates, and sends a summary of removed certificates to agents or administrators. For more information, see “Directory Update and Notification” on page 358.
	Both the Certificate Manager and Registration Manager provide this plug-in module.

## Certificate Renewal Notifications

When a certificate is about to expire, the owner of the certificate needs to renew it. Using the Jobs Scheduler, you can configure a Certificate Manager or Registration Manager to automatically send email-based renewal notices to users whose certificates are about to expire or have expired. You can also configure these subsystems to send one or more administrators or issuing agents a summary of users who have received these reminders.

### Plug-in Module for Automated Renewal Notifications

The plug-in module implementation for scheduling renewal notifications is identified as follows:

```
com.netscape.certsrv.jobs.RenewalNotificationJob
```

- In the CMS window, the implementation for this job class is identified as follows:

```
RenewalNotificationJob
```

- In the CMS configuration file, the implementation for this job class is identified as follows:

```
jobsScheduler.impl.RenewalNotificationJob.class=com.netscape
.certsrv.jobs.RenewalNotificationJob
```

This plug-in is a schedulable job. When an instance of the job is enabled, it checks for certificates that are about to expire in the internal database. When it finds one, it automatically emails the certificate's owner and continues sending email reminders for a configured period of time, or until the certificate is renewed. The job also collects a summary of all such renewal notifications and mails the summary to one or more agents or administrators.

The job determines the email address to which to send the notification using an email resolver, which you can customize. By default, the email address is found in the certificate itself or in the certificate's associated enrollment request.

The email notification message, as well as the summary message, are constructed using a template found in the configured directory. This directory has the following default location:

```
<server_root>/<instance_id>/emails
```

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

`<instance_id>` is the ID for this instance of Certificate Management System. You first specified this when you installed this server.

You can configure both the path and filenames of the template files for each job and modify the templates to customize the contents and appearance of the messages. Messages can be sent as HTML or plain text. For customization information, see “Customizing Notification Messages” on page 370.

For each instance of the `RenewalNotificationJob` class, you can configure the following:

- The schedule of times when the job will be run; see “Schedule for Executing Jobs” on page 363.
- How long before expiration the first notification will be sent.
- How long, after the certificate expires, notifications will continue to be sent if the certificate is not renewed.
- The sender of the notification messages (who will be notified of any delivery problems).
- The file location of the notification email template.
- The subject line of the notification message.

- How the email address for the notification is to be resolved.
- Whether a summary will be compiled and sent.

If a summary is to be sent, you can configure the following:

- The recipients of the summary message. These can be, for example, agents who need to know the status of user certificates.
- The sender of the summary message (who will be notified of any delivery problems).
- The file location of the summary message template.
- The file location of content and format of each item to be collected for the summary.
- The subject line of the summary message.

## Configurable Parameters

Figure 13.1 shows how the configurable parameters pertaining to the `RenewalNotificationJob` plug-in module are displayed in the CMS window.

Figure 13.1 Configurable parameters and values for the `RenewalNotificationJob` plug-in module

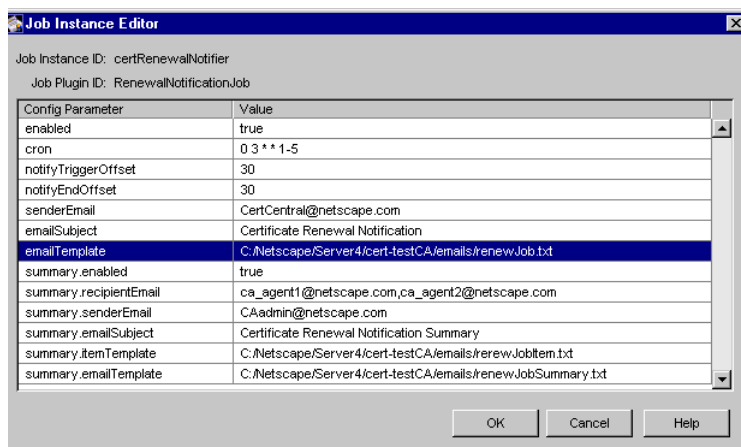


Table 13.2 gives details about each of these parameters.

Table 13.2 Configurable parameters for the renewal notification job and their values

Parameter name	Description
<code>enabled</code>	<p>Specifies whether the job is enabled or disabled. To enable the job, enter <code>true</code>; to disable the job, enter <code>false</code>.</p> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li>• <code>true</code></li> <li>• <code>false</code></li> </ul> <p>Object type: String</p>
<code>cron</code>	<p>Specifies the <code>cron</code> specification for when this job should be run. In other words, it specifies the time at which the Job Scheduler daemon thread should check the certificates for sending renewal notifications.</p> <p>Example: <code>03**1-5</code></p> <p>Permissible values: Must follow the convention specified in “Schedule for Executing Jobs” on page 363</p> <p>Object type: String</p>
<code>notifyTriggerOff set</code>	<p>Specifies how long (in days) before certificate expiration the first notification will be sent.</p> <p>Example: <code>30</code></p> <p>Permissible values: As applicable</p> <p>Object type: Integer</p>

Table 13.2 Configurable parameters for the renewal notification job and their values (Continued)

Parameter name	Description
<code>notifyEndOffset</code>	<p>Specifies how long (in days) after the certificate expire notifications will continue to be sent, if the certificate is not renewed.</p> <p>Example: 30</p> <p>Permissible values: As applicable</p> <p>Object type: Integer</p>
<code>senderEmail</code>	<p>Specifies the sender of the notification messages (who will be notified of any delivery problems).</p> <p>Example: <code>CertCentral@netscape.com</code></p> <p>Permissible values: Complete email address</p> <p>Object type: String</p>
<code>emailSubject</code>	<p>Specifies the subject line of the notification message.</p> <p>Example: <code>Certificate Renewal Notification</code></p> <p>Permissible values: Alphanumeric string of up to 255 characters</p> <p>Object type: String</p>
<code>emailTemplate</code>	<p>Specifies the path, including the filename, to the directory that contains the template to be used for formulating the message content.</p> <p>Example: <code>C:/Netscape/Server4/cert-testCA/emails/renewJob.txt</code></p> <p>Permissible values: Template file path, including the file name</p> <p>Object type: String</p>

Table 13.2 Configurable parameters for the renewal notification job and their values (Continued)

Parameter name	Description
<code>summary.enabled</code>	<p>Specifies whether a summary report of renewal notifications should be compiled and sent:</p> <ul style="list-style-type: none"> <li>• <code>true</code> - A summary should be sent.</li> <li>• <code>false</code> - A summary should not be sent.</li> </ul> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li>• <code>true</code></li> <li>• <code>false</code></li> </ul> <p>Object type: String</p>
<code>summary.recipientEmail</code>	<p>Specifies the recipients of the summary message. These can be, for example, agents who need to know the status of user certificates.</p> <p>Example: <code>ca_agent1@netscape.com, ca_agent2@netscape.com</code></p> <p>Permissible values: Full email addresses, separated by commas</p> <p>Object type: String</p>
<code>summary.senderEmail</code>	<p>Specifies the sender of the summary message (who will be notified of any delivery problems).</p> <p>Example: <code>CAadmin@netscape.com</code></p> <p>Permissible values: The full email address</p> <p>Object type: String</p>
<code>summary.emailSubject</code>	<p>Specifies the subject line of the summary message.</p> <p>Example: <code>Certificate Renewal Notification Summary</code></p> <p>Permissible values: Alphanumeric string of up to 255 characters</p> <p>Object type: String</p>



Table 13.2 Configurable parameters for the renewal notification job and their values (Continued)

Parameter name	Description
<code>summary.itemTemplate</code>	<p>Specifies the path, including the filename, to the directory that contains the template to be used for formulating the content and format of each item to be collected for the summary report (see the <code>summary.emailTemplate</code> parameter below). For details, see “Customizing Notification Messages” on page 370.</p> <p>Example: <code>C:/Netscape/Server4/cert-testCA/emails/renewJobItem.txt</code></p> <p>Permissible values: Template file path, including the file name</p> <p>Object type: String</p>
<code>summary.emailTemplate</code>	<p>Specifies the path, including the filename, to the directory that contains the template to be used for formulating the summary report. For details, see “Customizing Notification Messages” on page 370.</p> <p>Example: <code>C:/Netscape/Server4/cert-testCA/emails/renewJobSummary.txt</code></p> <p>Permissible values: Template file path, including the file name</p> <p>Object type: String</p>

## Notification of Request Queue Status

In addition to or instead of notifying agents of new requests, you might want to schedule a job that regularly notifies them of the status of the request queue. Such a job can check at a configured interval whether there are any *deferred* enrollment requests waiting for review. It can then send an email message to agents informing them of the number of requests waiting in the request queue for which they are responsible.

## Plug-in Module for Sending Notifications of Request Queue Status

The plug-in module provided out of the box for request-queue-status notifications is identified as follows:

```
com.netscape.certsrv.jobs.RequestInQJob
```

- In the CMS window, the implementation for this job class is identified as follows:

```
RequestInQJob
```

- In the CMS configuration file, the implementation for this job class is identified as follows:

```
jobsScheduler.impl.RequestInQJob.class=com.netscape.certsrv.  
jobs.RequestInQJob
```

This plug-in is a schedulable job. When an instance of the job is enabled, it gets activated at the configured interval and checks the status of the request queue. If any deferred enrollment requests are waiting in the queue, the job constructs an email message summarizing its findings and sends it to the specified agents.

The job constructs the summary message by using a template located in a configured directory. This directory has the following default location:

```
<server_root>/<instance_id>/emails
```

**<server\_root>** is the directory where the CMS binaries are kept. You first specified this directory during installation.

**<instance\_id>** is the ID for this instance of Certificate Management System. You first specified this when you installed this server.

You can configure the path and filename of the template file for each job and modify the templates to customize the contents and appearance of the messages. Messages can be sent as HTML or plain text.

For each instance of the `RequestInQJob` class, you can configure the following:

- The subsystem, Certificate Manager or Registration Manager, that should use this job.
- The schedule of times when the job will be run; see “Schedule for Executing Jobs” on page 363.
- The sender of the notification messages (who will be notified of any delivery problems).
- The file location of the notification email template.
- The subject line of the notification message.
- The email addresses of message recipients; these should be subsystem agents whose task it is to review manual enrollment requests.

## Configurable Parameters

Figure 13.2 shows how the configurable parameters pertaining to the `RequestInQJob` plug-in module are displayed in the CMS window.

Figure 13.2 Configurable parameters and values for the `RequestInQJob` plug-in module

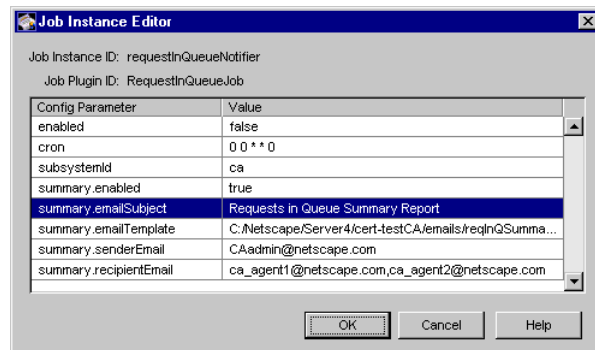


Table 13.4 gives details for each of these parameters.

Table 13.3 Configurable parameters for the request-in-queue notification job and their values

Parameter name	Description
<code>enabled</code>	<p>Specifies whether the job is enabled or disabled. To enable the job, enter <code>true</code>; to disable the job, enter <code>false</code>.</p> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li>• <code>true</code></li> <li>• <code>false</code></li> </ul> <p>Object type: String</p>
<code>cron</code>	<p>Specifies the <code>cron</code> specification for when this job should be run. This is the time at which the Job Scheduler daemon thread checks the queue for pending requests.</p> <p>Example: <code>00**0</code></p> <p>Permissible values: Must follow the convention specified in “Schedule for Executing Jobs” on page 363</p> <p>Object type: String</p>
<code>subsystemid</code>	<p>Specifies the subsystem that this job is for.</p> <ul style="list-style-type: none"> <li>• <code>ca</code> - The job is for the Certificate Manager.</li> <li>• <code>ra</code> - The job is for the Registration Manager.</li> </ul> <p>Example: <code>ca</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li>• <code>ca</code></li> <li>• <code>ra</code></li> </ul> <p>Object type: String</p>

Table 13.3 Configurable parameters for the request-in-queue notification job and their values (Continued)

Parameter name	Description
<code>summary.enabled</code>	<p>Specifies whether a summary of the job accomplished should be compiled and sent:</p> <ul style="list-style-type: none"> <li>• <code>true</code> - A summary should be sent.</li> <li>• <code>false</code> - A summary should not be sent.</li> </ul> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li>• <code>true</code></li> <li>• <code>false</code></li> </ul> <p>Object type: String</p>
<code>summary.emailSubject</code>	<p>Specifies the subject line of the summary message.</p> <p>Example: <code>Summary Report of Requests in the Agent Queue</code></p> <p>Permissible values: Alphanumeric string of up to 255 characters</p> <p>Object type: String</p>
<code>summary.emailTemplate</code>	<p>Specifies the path, including the filename, to the directory that contains the template to be used for formulating the summary report. For details, see “Customizing Notification Messages” on page 370.</p> <p>Example: <code>C:/Netscape/Server4/cert-testCA/emails/reqInQJobSummary.txt</code></p> <p>Permissible values: Template file path, including the file name</p> <p>Object type: String</p>

Table 13.3 Configurable parameters for the request-in-queue notification job and their values (Continued)

Parameter name	Description
<code>summary.senderEmail</code>	<p>Specifies the sender of the notification message (who should be notified of any delivery problems).</p> <p>Example: CAadmin@netscape.com</p> <p>Permissible values: The full email address</p> <p>Object type: String</p>
<code>summary.recipientEmail</code>	<p>Specifies the recipients of the summary message. These should be, for example, agents who need to process pending requests.</p> <p>Example: ca_agent1@netscape.com, ca_agent2@netscape.com</p> <p>Permissible values: Full email addresses, separated by commas</p> <p>Object type: String</p>

## Directory Update and Notification

Certificate Management System doesn't automatically remove expired certificates from the publishing directory. If you configure a Certificate Manager or Registration Manager to publish certificates to an LDAP directory, over time the directory will contain expired certificates. To help you remove expired certificates from the directory, both the Certificate Manager and Registration Manager come with a plug-in module that allows you to create a schedulable job that periodically removes (or unpublishes) certificates that have expired. When the directory has been updated, the job can collect a summary report of the certificates that have been removed and send it to people who need to have this information. Typically, you would want to send this notification to certificate issuing agents or the administrator of the publishing directory.

Note that the job automates removal of expired certificates from the directory. You can also remove expired certificates manually following the instructions in "Manually Updating Certificate Information in the Directory" on page 520.

## Plug-in Module for Removing Expired Certificates from the Directory

The plug-in module implementation for scheduling removal of expired certificates from the directory is identified as follows:

```
com.netscape.certsrv.jobs.UnpublishExpiredJob
```

- In the CMS window, the implementation for this job class is identified as follows:

```
UnpublishExpiredJob
```

- In the CMS configuration file, the implementation for this job class is identified as follows:

```
jobsScheduler.impl.ExpiredUnpublishJob.class=com.netscape.certsrv.jobs.UnpublishExpiredJob
```

This plug-in is a schedulable job. When an instance of the job is enabled, it gets activated at the configured interval and checks for certificates that have expired and are still marked as *published* in the internal database. The job connects to the publishing directory and deletes these certificates; it then marks these certificates as *unpublished* in the internal database. The job also collects a summary of expired certificates that it deleted and mails the summary to one or more agents or administrators as specified by the configuration.

The job constructs the summary message by using a template located in a configured directory. This directory has the following default location:

```
<server_root>/<instance_id>/emails
```

**<server\_root>** is the directory where the CMS binaries are kept. You first specified this directory during installation.

**<instance\_id>** is the ID for this instance of Certificate Management System. You first specified this when you installed this server.

You can configure the path and filename of the template file for each job. You can also modify the templates to customize the contents and appearance of the messages; see “Customizing Message Templates” on page 374.

Messages can be sent as HTML or plain text.

For each instance of the `UnpublishExpiredJob` class, you can configure the following:

- The schedule of times when the job will be run; see “Schedule for Executing Jobs” on page 363.
- Whether a summary will be compiled and sent.

If a summary is to be sent, you can configure the following:

- The recipients of the summary message. These can be, for example, administrators who are responsible for the publishing directory.
- The sender of the summary message (who will be notified of any delivery problems).
- The file location a of the summary message template.
- The file location of content and format of each item to be collected for the summary.
- The subject line of the summary message.

**Configurable Parameters**

Figure 13.3 shows how the configurable parameters pertaining to the `UnpublishExpiredJob` plug-in module are displayed in the CMS window.

**Figure 13.3** Configurable parameters and values for the `UnpublishExpiredJob` plug-in module

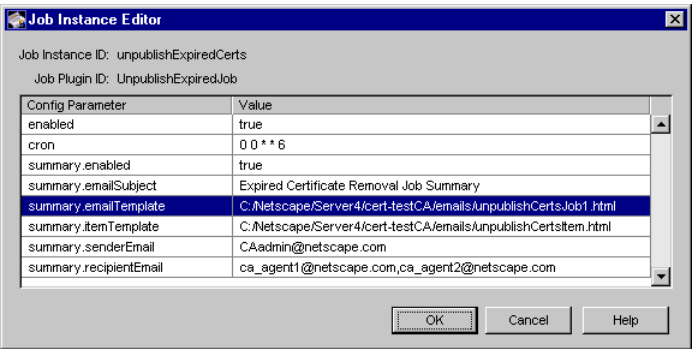




Table 13.4 gives details for each of these parameters.

**Table 13.4 Configurable parameters for the unpublish expired certificates job and their values**

Parameter name	Description
<code>enabled</code>	<p>Specifies whether the job is enabled or disabled. To enable the job, enter <code>true</code>; to disable the job, enter <code>false</code>.</p> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li>• <code>true</code></li> <li>• <code>false</code></li> </ul> <p>Object type: String</p>
<code>cron</code>	<p>Specifies the <code>cron</code> specification for when this job should be run. This is the time at which the Job Scheduler daemon thread checks the certificates for removing expired certificates from the publishing directory.</p> <p>Example: <code>00**6</code></p> <p>Permissible values: Must follow the convention specified in “Schedule for Executing Jobs” on page 363</p> <p>Object type: String</p>
<code>summary.enabled</code>	<p>Specifies whether a summary of the certificates removed by the job should be compiled and sent:</p> <ul style="list-style-type: none"> <li>• <code>true</code> - A summary should be sent.</li> <li>• <code>false</code> - A summary should not be sent.</li> </ul> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li>• <code>true</code></li> <li>• <code>false</code></li> </ul> <p>Object type: String</p>

Table 13.4 Configurable parameters for the unpublish expired certificates job and their values (Continued)

Parameter name	Description
<code>summary.emailSubject</code>	<p>Specifies the subject line of the summary message.</p> <p>Example: Expired Certificate Removal Job Summary</p> <p>Permissible values: Alphanumeric string of up to 255 characters</p> <p>Object type: String</p>
<code>summary.emailTemplate</code>	<p>Specifies the path, including the filename, to the directory that contains the template to be used for formulating the summary report. For details, see “Customizing Notification Messages” on page 370.</p> <p>Example: C:/Netscape/Server4/cert-testCA/emails/unpublishCertsJobSummary.html</p> <p>Permissible values: Template file path, including the file name</p> <p>Object type: String</p>
<code>summary.itemTemplate</code>	<p>Specifies the path, including the filename, to the directory that contains the template to be used for formulating the content and format of each item to be collected for the summary report (see the <code>summary.emailTemplate</code> parameter above).</p> <p>Example: C:/Netscape/Server4/cert-testCA/emails/unpublishCertsJobItem.txt</p> <p>Permissible values: Template file path, including the file name</p> <p>Object type: String</p>
<code>summary.senderEmail</code>	<p>Specifies the sender of the summary message (who should be notified of any delivery problems).</p> <p>Example: CAadmin@netscape.com</p> <p>Permissible values: The full email address</p> <p>Object type: String</p>

Table 13.4 Configurable parameters for the unpublish expired certificates job and their values (Continued)

Parameter name	Description
<code>summary.recipientEmail</code>	Specifies the recipients of the summary message. These can be, for example, agents who need to know the status of user certificates.  Example: <code>cert_agent1@netscape.com, cert_agent2@netscape.com</code>  Permissible values: Complete email addresses, separated by commas  Object type: String

## Schedule for Executing Jobs

The Job Scheduler uses a variation of the Unix `crontab` entry format to specify dates and times for checking the job queue and executing jobs. As shown in Table 13.5, the time entry format consists of five fields (the sixth field specified for the Unix `crontab` is not used by the Job Scheduler). Values are separated by spaces or tabs.

Table 13.5 Time format for scheduling jobs

Field	Value
Minute	0-59
Hour	0-23
Day of month	1-31
Month of year	1-12
Day of week	0-6 (where 0=Sunday)

Each field can contain either a single integer or a pair of integers separated by a hyphen or dash (-) to indicate an inclusive range. To specify all legal values, a field can contain an asterisk rather than an integer. Day fields can contain a comma-separated list of values.

For example, the following time entry specifies every hour at 15 minutes (1:15, 2:15, 3:15 and so on):

```
15 * * * *
```

The following example specifies a job execution time of noon on April 12:

```
0 12 12 4 *
```

The day-of-month and day-of-week fields can contain a comma-separated list of values to specify more than one day. If both day fields are specified, the specification is inclusive; that is, the day of the month is not required to fall on the day of the week to be valid. For example, the following entry specifies a job execution time of midnight on the first and fifteenth of every month, *and* on every Monday:

```
0 0 1,15 * 1
```

To specify one day type without the other, use an asterisk in the other day field. For example, the following entry specifies a job execution time of 3:15 a.m. on every weekday morning:

```
15 3 * * 1-5
```

## Event-Driven Notifications

You can configure a Certificate Manager or Registration Manager to send email notifications automatically to end entities, agents, or administrators when certain events occur. Unlike jobs that are executed at a preconfigured schedule by the Job Scheduler component, these notifications are event driven—that is, whenever the specified event occurs, the server notifies the configured user.

Notifiable events include the following:

- Notifications of Certificate Issuance to End Entities

End entities are notified by email that a requested certificate has been issued. This is an event-driven notification.

- Notification of New Request in Queue

Agents are notified by email that a request has been added to the request queue. Alternatively (or in addition) a schedulable job can notify agents at regular intervals of the current state of the request queue; see “Notification of Request Queue Status” on page 353.

## Notifications of Certificate Issuance to End Entities

You can configure the Certificate Manager or Registration Manager to send a notification message to users who have been issued certificates in response to enrollment requests. This message normally includes information about the issued certificate and instructions for importing the certificate into the user's client.

This kind of notification involves a listener class in the subsystem that registers an interest in an appropriate event, in this case successful completion of an enrollment request. In the CMS configuration, this listener class for a Certificate Manager is defined as `ca.notification.certIssued` and for the Registration Manager it is defined as `ra.notification.certIssued`.

When a certificate is issued, the listener builds a notification message based on a configured template and sends it to an email address that it determines by using an email resolver. By default, the email address is found in the certificate itself or in the certificate's associated enrollment request.

The template that the listener uses to construct the email notification message is located in the configured directory. This directory has the following default location:

```
<server_root>/<instance_id>/emails
```

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

`<instance_id>` is the ID for this instance of Certificate Management System. You first specified this when you installed this server.

You can configure both the path and filename of the template file. You can also modify the template to customize the contents and appearance of the messages; see “Customizing Message Templates” on page 374.

Messages can be sent as HTML or plain text.

For the `certIssued` listener, you can configure the following:

- Whether the listener is enabled.

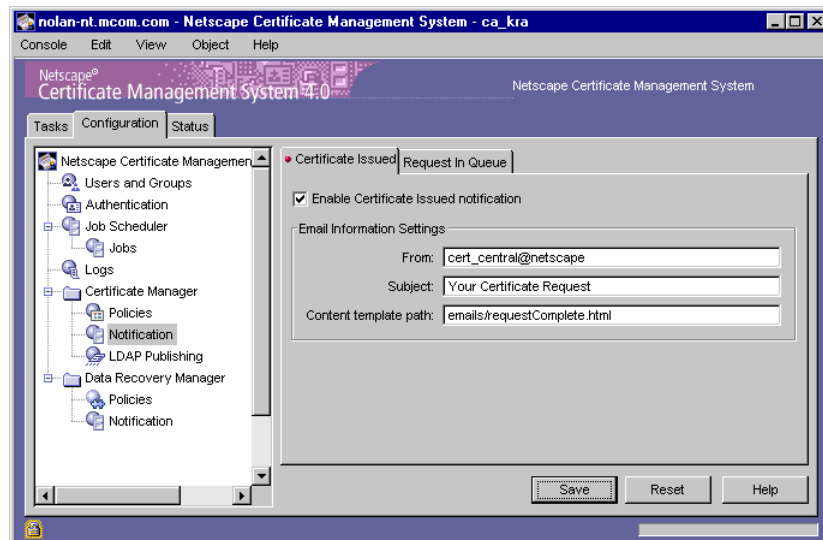
- The sender of the notification messages (who will be notified of any delivery problems).
- The location of the notification email template.
- The subject line of the notification message.

## Configuring a Subsystem to Send Notifications to End Entities

To configure a Certificate Manager or Registration Manager to send certificate-issuance notifications to end entities:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, select the subsystem, then click Notification. (The figure below shows the Certificate Manager’s notification feature; the Registration Manager also has a similar feature.)

The Certificate Issued tab appears.



4. To enable the notification feature, check the “Enable Certificate Issued notification” option.

5. In the Email Information Settings section, enter information as appropriate:

**From.** Type the sender's full email address (this is the person who should be notified of any delivery problems).

**Subject.** Type the subject title for the notification.

**Content template path.** Type the path, including the filename, to the directory that contains the template to be used for formulating the message content.

6. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

## Notification of New Request in Queue

When a *deferred* end-entity request enters the request queue of a Certificate Manager or Registration Manager, agents assigned to manage the queue must review the request and reject or accept it. To help ensure that an agent processes the request in a timely manner, you can configure the subsystem to notify agents whenever a new request gets added to the request queue.

This kind of notification involves a listener class in the subsystem that registers an interest in an appropriate event, in this case the addition of a request to the request queue. In the CMS configuration, this listener class is identified as follows:

```
ca.notification.requestInQ
```

When a request is added to the queue, the listener builds a notification message based on a configured template and sends it to one or more agents' email addresses as configured.

The template that the listener uses to construct the email notification message is located in the configured directory. This directory has the following default location:

```
<server_root>/<instance_id>/emails
```

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

`<instance_id>` is the ID for this instance of Certificate Management System. You first specified this when you installed this server.

You can configure both the path and filename of the template file. You can also modify the template to customize the contents and appearance of the messages; see “Customizing Message Templates” on page 374.

For the `requestInQ` event listener, you can configure the following:

- Whether the listener is enabled.
- The sender of the notification messages (who will be notified of any delivery problems).
- The location of the notification email template.
- The subject line of the notification message.
- The email addresses of message recipients; these should be subsystem agents whose task it is to review *deferred* enrollment requests.

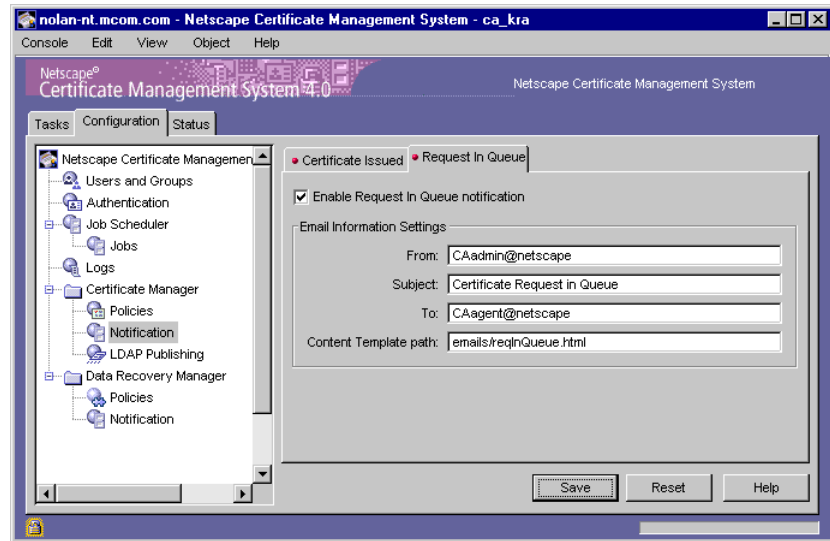
## Configuring a Subsystem to Send Request Queue Notifications

To configure a Certificate Manager or Registration Manager to send email notifications to its agents:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, select the subsystem, and then click Notification. (The figure below shows the Certificate Manager’s notification feature; the Registration Manager also has a similar feature.)



4. In the right pane, click Request In Queue.



5. To enable the notification feature, check the “Enable Request In Queue notification” option.
6. Enter information as appropriate:

**From.** Type the sender’s full email address (this is the person who should be notified of any delivery problems).

**Subject.** Type the subject title for the notification—for example, “End Entity Request in Queue.”

**To.** Type the recipient’s full email address (this is the person who will check the queue). You can specify more than one recipient; separate email addresses by commas.

**Content template path.** Type the path, including the filename, to the directory that contains the template to be used for formulating the message content.

7. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

# Customizing Notification Messages

Notification and summary email messages are constructed using templates located in the `emails` directory of a CMS instance. This directory has the following default location:

```
<server_root>/<instance_id>/emails/...
```

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

`<instance_id>` is the ID for this instance of Certificate Management System. You first specified this when you installed this server.

Both text and HTML templates are included by default. They are listed in Table 13.6 and Table 13.7.

## Templates for Event-Triggered Notifications

Table 13.6 lists the default template files provided for formulating event-triggered-notification messages. You can change the name of these files as applicable; be sure to make the appropriate changes to the configuration (see the “Content template file” field on page 367 and page 369).

Tokens, which you can use as variables in the body of the message, are defined for each template enabling you to customize the message; the token is replaced by its current variable value in the constructed message. For details, see “Customizing Message Templates” on page 374.

**Table 13.6** Default templates for event-triggered notifications

Filename	Description
<code>certIssued_CA</code>	Template for the Certificate Manager to send plain-text notifications to end entities upon issuance of certificates.

Table 13.6 Default templates for event-triggered notifications (Continued)

Filename	Description
certIssued_CA.html	Template for the Certificate Manager to send HTML notifications to end entities upon issuance of certificates.
certIssued_RA	Template for the Registration Manager to send plain-text notifications to end entities upon issuance of certificates.
certIssued_RA.html	Template for the Registration Manager to send HTML notifications to end entities upon issuance of certificates.
reqInQueue	Template for the Certificate Manager or Registration Manager to send plain-text notifications to agents when a request enters the queue.
reqInQueue.html	Template for the Certificate Manager or Registration Manager to send plain-text notifications to agents when a request enters the queue.

Note that in the CMS configuration, template files for event-triggered notifications are identified as follows:

```
<subsystem>.notification.<notification_name>.emailTemplate=
<template_file_path>
```

```
<subsystem>.notification.<notification_name>.emailTemplate=
<template_file_path>
```

`<subsystem>` specifies the prefix that identifies the subsystem to which the configuration parameter belongs—`ca` for the Certificate Manager and `ra` for the Registration Manager.

`<notification_name>` specifies the name of the event-triggered notification—`certIssued` for the certificate issuance notifications to end entities and `requestInQ` for the request in queue notifications to agents.

`<template_file_path>` specifies the path, including the filename, to the directory that contains the template to be used for formulating the message content.

## Templates for Summary Notifications

Table 13.7 lists the default template files for formulating the notification messages that summarize jobs that were executed by the Job Scheduler component of a Certificate Manager or Registration Manager. You can change the name of these files as applicable; be sure to make the appropriate changes to the configuration.

For summaries, a separate template is used to format the entry for each item in the summary. The item entries are then added to a table in the summary message.

Tokens, which you can use as variables in the body of the message, are defined for each templates enabling you to customize the message; the token is replaced by its current variable value in the constructed message. For details, see “Customizing Message Templates” on page 374.

Table 13.7 Default templates for summary notifications

Filename	Description
Templates for UnpublishExpiredJob module	
ExpiredUnpublishJob	Template for formulating the summary report or table that summarizes removal of expired certificates from the directory.
ExpiredUnpublishJobItem	Template for formatting the items to be included in the summary table, which is constructed using the <code>ExpiredUnpublishJob</code> template.
Templates for RequestInQueueJob module	

Table 13.7 Default templates for summary notifications (Continued)

Filename	Description
<ul style="list-style-type: none"><li>• <code>riqlItem.html</code></li></ul>	Template for formatting the items to be included in the summary table, which is constructed using the <code>riqlSummary.html</code> template.
<ul style="list-style-type: none"><li>• <code>riqlSummary.html</code></li></ul>	Template for formulating the summary report or table that summarizes how many requests are pending in the agent queue of a Certificate Manager or Registration Manager.
Templates for <code>RenewalNotificationJob</code> module	
<ul style="list-style-type: none"><li>• <code>rnJob1.txt</code></li></ul>	Template for formulating the message content to be sent to end entities to inform them that their certificates are about to expire and that they should renew their certificates before expiration.
<ul style="list-style-type: none"><li>• <code>rnJob1Item.txt</code></li></ul>	Template for formatting the items to be included in the summary report, which is constructed using the <code>rnJob1Summary.txt</code> template.
<ul style="list-style-type: none"><li>• <code>rnJob1Summary.txt</code></li></ul>	Template for formulating the summary report to be sent to agents and administrators.

Note that in the CMS configuration, template files for schedulable jobs are identified as follows:

```
jobsScheduler.job.<job_name>.summary.emailTemplate=  
<template_file_path>
```

```
jobsScheduler.job.<job_name>.summary.itemTemplate=  
<template_file_path>
```

`<job_name>` specifies the job instance name. For a list of job instances created by default, see Table 14.1 on page 392.

`<template_file_path>` specifies the path, including the filename, to the directory that contains the template to be used for formulating the message content.

## Customizing Message Templates

You can modify the templates to customize the contents and appearance of messages. The message body can contain HTML or plain text. In the body of the message, you can use tokens or keywords as variables. A token is indicated by the dollar character (\$) and is replaced by its current variable value in the constructed message. Different tokens are available for each job or notification class. These are listed in “Tokens Available in Message Templates” on page 374.

For example, a certificate-issuance-notification message can make use of tokens as follows:

### CERTIFICATE ISSUANCE NOTIFICATION

Your certificate request (\$RequestId) has been processed successfully.  
Details of your certificate are as follows:

Serial Number= \$SerialNumber

SubjectDN= \$SubjectDN

IssuerDN= \$IssuerDN

Validity Period= \$NotBefore - \$NotAfter

To get your certificate, please follow this URL:

[https://\\$HttpHost:\\$HttpPort/getCertFromRequest?requestId=\\$RequestId](https://$HttpHost:$HttpPort/getCertFromRequest?requestId=$RequestId)

If you have any questions or problems, please send an email to  
[cert\\_central@netscape.com](mailto:cert_central@netscape.com).

Thank you.

## Tokens Available in Message Templates

This section explains the tokens provided in the templates used by the default job plug-in and event-triggered notification modules to formulate notification messages.

- Tokens for Certificate Issuance Notifications to End Entities
- Tokens for Renewal Notification Messages

- Tokens for Request In Queue Notification Messages
- Tokens for Directory Update Notification Messages

## Tokens for Certificate Issuance Notifications to End Entities

Table 13.8 lists the tokens that are available in the message templates provided for formulating the content of email notifications to end entities; a Certificate Manager or Registration Manager can send these notifications upon issuance of the certificates they requested.

Table 13.8 Tokens defined in templates used for certificate-issuance notifications

Token	Description
\$HttpHost	Specifies the fully qualified host name of the Certificate Manager or Registration Manager to which end entities should connect to retrieve their certificates. (This token enables you to construct the URL from which end entities can download their certificates; see the example in “Customizing Message Templates” on page 374.)
\$HttpPort	Specifies the port number at which the Certificate Manager or Registration Manager is listening to end-entity requests. (This token enables you to construct the URL from which end entities can download their certificates; see the example in “Customizing Message Templates” on page 374.)
\$InstanceID	<div>Specifies the ID assigned to the subsystem that sent this notification.</div> <ul style="list-style-type: none"><li>• If the notification is sent by a Certificate Manager, this will be <code>ca</code>.</li><li>• If the notification is sent by a Registration Manager, this will be <code>ra</code>.</li></ul>
\$IssuerDN	Specifies the distinguished name of the CA that issued the certificate.

Table 13.8 Tokens defined in templates used for certificate-issuance notifications (Continued)

Token	Description
\$NotAfter	Specifies the NotAfter attribute.
\$NotBefore	Specifies the NotBefore attribute.
\$RequestId	Specifies the request ID.
\$SerialNumber	Specifies the serial number of the certificate that has been issued.
\$SubjectDN	Specifies the distinguished name of the certificate subject.

## Tokens for Renewal Notification Messages

This section lists the tokens that are available in the message templates for instances of the `RenewalNotificationJob` class or plug-in module.

Table 13.9 lists the tokens that you can use for formulating this job's summary report. You can customize the content and format of the items in the report by using the tokens defined in Table 13.10.

Table 13.9 Tokens for the renewal-notification job's summary report

Token	Description
\$ExecutionTime	Specifies the time the job (instance) was run.
\$InstanceID	Specifies the name of the job instance.
\$SummaryItemList	Specifies the list of items in the summary notification. Each item corresponds to a certificate the job detects for renewal.
\$SummaryTotalfailure	Specifies the total number of items in the summary report that failed.



Table 13.9 Tokens for the renewal-notification job's summary report (Continued)

Token	Description
<code>\$SummaryTotalNum</code>	Specifies the total number of items (certificates that require to be renewed) in the summary report.
<code>\$SummaryTotalSuccess</code>	Specifies how many of the total number of items in the summary report succeeded.

Table 13.10 lists the tokens for the inner list items.

Table 13.10 Tokens for items in renewal-notification job's summary report

Token	Description
<code>\$CertType</code>	Specifies the type of certificate—whether SSL client ( <code>client</code> ), SSL server ( <code>server</code> ), Registration Manager's signing certificate ( <code>ra</code> ), Certificate Manager's CA signing certificate ( <code>ca</code> ), router certificate ( <code>Cisco-router</code> ), or other ( <code>other</code> ).
<code>\$HttpHost</code>	Specifies the fully qualified host name of the Certificate Manager or Registration Manager to which end entities should connect to renew their certificates. (The token enables you to construct the URL which end entities use to renew their certificates; see the example in “Customizing Message Templates” on page 374.)
<code>\$HttpPort</code>	Specifies the port number at which the Certificate Manager or Registration Manager is listening to certificate-renewal requests from end entities. (The token enables you to construct the URL which end entities use to renew their certificates; see the example in “Customizing Message Templates” on page 374.)
<code>\$IssuerDN</code>	Specifies the distinguished name of the certificate issuer.
<code>\$NotAfter</code>	Specifies the <code>NotAfter</code> attribute.
<code>\$NotBefore</code>	Specifies the <code>NotBefore</code> attribute.

Table 13.10 Tokens for items in renewal-notification job's summary report (Continued)

Token	Description
\$RequestorEmail	Specifies the requestor's email address.
\$RequestType	Specifies the request type—whether it is a certificate enrollment, certificate renewal, certificate revocation, key archival, or key recovery request.
\$SerialNumber	Specifies the serial number of the certificate.
\$Status	Specifies whether the operation failed or succeeded.
\$SubjectDN	Specifies the distinguished name of the certificate subject.

## Tokens for Request In Queue Notification Messages

Table 13.11 lists the tokens that you can use for formulating the content of the RequestInQueueJob job's summary report.

Table 13.11 Tokens for the request-in-queue job's summary report

Token	Description
\$InstanceID	Specifies the ID assigned to the subsystem that sent this notification. <ul style="list-style-type: none"><li>• If the notification is sent by a Certificate Manager, this will be <code>ca</code>.</li><li>• If the notification is sent by a Registration Manager, this will be <code>ra</code>.</li></ul>
\$ExecutionTime	Specifies the time the job (instance) was run.
\$SummaryTotalNum	Specifies the total number of items (certificate requests that are pending in the queue) in the summary report.

## Tokens for Directory Update Notification Messages

This section lists the tokens that are available in summary message templates for instances of the `UnpublishExpiredJob` class or plug-in module.

Table 13.12 lists the tokens that are available for this job's summary report. You can customize the content and format of the items in the report by using the tokens defined in Table 13.13.

**Table 13.12** Tokens for the unpublish-expired job's summary report

Token	Description
<code>\$InstanceID</code>	Specifies the name of the job instance that generated this summary report.
<code>\$ExecutionTime</code>	Specifies the time the job (instance) was run.
<code>\$SummaryItemList</code>	Specifies the list of items in the summary notification. Each item corresponds to a certificate the job detects for removal from the publishing directory.
<code>\$SummaryTotalFailure</code>	Specifies the total number of items in the summary report that failed.
<code>\$SummaryTotalNum</code>	Specifies the total number of items (certificates to be removed from the directory) in the summary report.
<code>\$SummaryTotalSuccess</code>	Specifies how many of the total number of items in the summary report succeeded.

Table 13.13 lists the tokens for the inner list items.

**Table 13.13** Tokens for items in the unpublish-expired job's summary report

Token	Description
<code>\$CertType</code>	Specifies the type of certificate—whether SSL client ( <code>client</code> ), SSL server ( <code>server</code> ), Registration Manager's signing certificate ( <code>ra</code> ), Certificate Manager's CA signing certificate ( <code>ca</code> ), router certificate ( <code>Cisco-router</code> ), or other ( <code>other</code> ).

Table 13.13 Tokens for items in the unpublish-expired job's summary report (Continued)

Token	Description
<code>\$IssuerDN</code>	Specifies the distinguished name of the certificate issuer.
<code>\$NotAfter</code>	Specifies the <code>NotAfter</code> attribute.
<code>\$NotBefore</code>	Specifies the <code>NotBefore</code> attribute.
<code>\$RequestorEmail</code>	Specifies the requestor's email address.
<code>\$SerialNumber</code>	Specifies the serial number of the certificate.
<code>\$Status</code>	Specifies whether the operation failed or succeeded.
<code>\$SubjectDN</code>	Specifies the distinguished name of the certificate subject.

# Configuring Jobs

Netscape Certificate Management System (CMS) provides a customizable Job Scheduler component that supports various mechanisms for scheduling cron jobs. This chapter explains how to configure Certificate Management System to use specific job plug-in modules for accomplishing jobs. The chapter also shows how plug-in implementations and configured instances for various job items appear in the configuration file.

Before reading this chapter, you should have read the chapter “Introduction to Job Scheduling and Notifications” on page 345. In particular, you should be familiar with the various job plug-in modules that come with Certificate Management System. If you are not, see “Built-in Job Plug-in Modules” on page 346.

The chapter has the following sections:

- Job Management (page 382)
- Job Plug-in Implementation and Instance (page 387)
- Managing Jobs (page 387)
- Managing Job Plug-in Modules (page 396)

# Job Management

You can manage jobs in two ways:

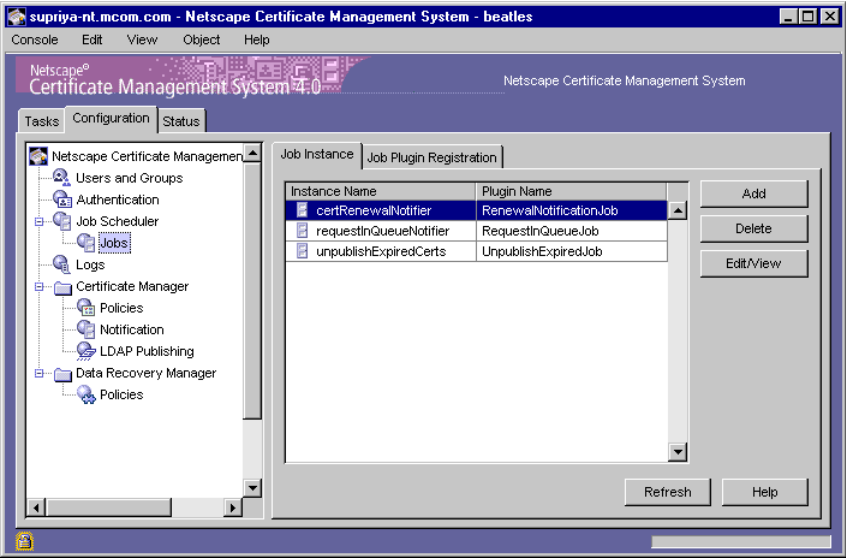
- By making changes to the job-specific parameters from the CMS window. See “Job Management from the CMS Window” on page 382.
- By editing the parameters in the configuration file. See “Job Scheduler Parameters in the Configuration File” on page 385.

The recommended method is to use the CMS window.

## Job Management from the CMS Window

Figure 14.1 shows the CMS window, which provides the required user interface to support job management.

Figure 14.1 Job Scheduler information in the CMS window



In the CMS window you will find a single `Job Scheduler` object, within which is another object identified as `Jobs`. The `Jobs` object represents the job plug-in implementations and instances currently recognized by this instance of Certificate Management System; the `Job Scheduler` object represents the schedule for the configured jobs.

From the CMS window you can accomplish the following operations:

- Configuration of currently registered job plug-ins
- Registration of new (custom) job plug-ins

The sections that follow describe the parts of the window from which you carry out these operations.

## Job Instance Tab

The Job Instance tab lists jobs that are currently scheduled for the server, so that you can manage them at a single place. From this tab you can perform the following operations:

**Add.** The add operation shows a list of registered job plug-in modules from which you can select the one you want to configure. You configure an instance of the selected module with the help of the job instance editor (see “Job Instance Editor” on page 384). When you save the changes, Certificate Management System creates the instance and displays it in the list of jobs. For instructions on adding new jobs to the CMS configuration, see “Adding a Job” on page 387.

**Delete.** The delete operation allows you to remove unwanted jobs from the CMS configuration. For instructions on deleting jobs, see “Deleting a Job” on page 391.

**Edit/View.** The edit operation allows you to view and modify configuration parameter values of currently configured jobs. You modify the parameter values with the help of the job instance editor (see “Job Instance Editor” on page 384). For instructions on modifying jobs, see “Modifying a Job” on page 391.

## Job Instance Editor

The job instance editor is designed to be generic. Its simple graphical interface enables you to create new job instances and modify the configuration of individual jobs. When you are *adding* a new job, the editor shows the configuration parameters pertaining to the plug-in module you selected. When you are *modifying* a job, the editor shows the configuration parameters pertaining to the job you selected.

All configurable parameters are displayed in the form of a table with two columns and multiple rows, each parameter occupying a row in the table. The left column lists the names of the configurable parameters; the right column is designated for entering the appropriate values. The ordering of the configurable parameters is irrelevant unless it is defined by the job plug-in implementation.

The job instance editor provides normal save, cancel, and help functionality. You can specify *names* for jobs, but only at the time of adding new ones; you cannot change names later.

## Job Plugin Registration Tab

The Job Plugin Registration tab lists the currently registered job plug-in modules for the selected CMS instance and gives you access to the window from which you can register new modules. On this tab you will find the names of registered plug-in modules listed on the left and the path to the Java class that implements the plug-in module listed on the right.

You can perform the following operations from this tab:

**Register.** This operation allows you to register a new job plug-in module. You do this with the help of the job registration editor (see “Job Plugin Registration Editor” on page 385).

When you save the changes, Certificate Management System loads the plug-in module implementation for the job and displays it in the list of currently registered plug-in modules. For instructions on registering new modules, see “Registering a Job Plug-in Module” on page 396.

**Delete.** This operation allows you to remove unwanted job plug-in modules from the CMS framework. For instructions on deleting modules, see “Deleting a Job Scheduler Plug-in” on page 398.



## Job Plugin Registration Editor

The job plug-in registration editor allows you to register new job plug-in modules in the CMS framework. Registering a new module involves specifying its name and the full name of the Java class that implements the module.

For example, you can add a job implementation named as follows:

```
com.netscape.jobscheduler.unpublishUserCert
```

## Job Scheduler Parameters in the Configuration File

The sample in Figure 14.2 shows how Job Scheduler-specific information appears in the configuration file. Keep the following points in mind:

- All job-specific information, such as registered job implementations and configured instances, appears in the Job Scheduler section of the configuration file.
- Each registered job plug-in is identified by its implementation name.
- Each job (or configured instance of a job plug-in module) is identified by the name you specified when creating it.
- You can create as many instances of an implementation as you like; each instance must have a unique name. For details, see “Job Plug-in Implementation and Instance” on page 387.

Figure 14.2 Job-specific parameters in the configuration file

```

jobsScheduler._000=##
jobsScheduler._001=## jobScheduler
jobsScheduler._002=##
jobsScheduler.enabled=false
jobsScheduler.interval=1
jobsScheduler.impl.RenewalNotificationJob.class=com.netscape.certsrv.jobs.RenewalNotificationJob
jobsScheduler.impl.RequestInQueueJob.class=com.netscape.certsrv.jobs.RequestInQueueJob
jobsScheduler.impl.UnpublishExpiredJob.class=com.netscape.certsrv.jobs.UnpublishExpiredJob
jobsScheduler.job.substores=unpublishExpiredCerts

jobsScheduler.job.certRenewalNotifier.cron=0 3 * * 1-5
jobsScheduler.job.certRenewalNotifier.emailSubject=Certificate Renewal Notification
jobsScheduler.job.certRenewalNotifier.emailTemplate=C:/Netscape/Server4/cert-testCA/emails/renewJob.bt
jobsScheduler.job.certRenewalNotifier.enabled=true
jobsScheduler.job.certRenewalNotifier.notifyEndOffset=30
jobsScheduler.job.certRenewalNotifier.notifyTriggerOffset=30
jobsScheduler.job.certRenewalNotifier.pluginName=RenewalNotificationJob
jobsScheduler.job.certRenewalNotifier.senderEmail=CertCentral@netscape.com
jobsScheduler.job.certRenewalNotifier.summary.emailSubject=Certificate Renewal Notification Summary
jobsScheduler.job.certRenewalNotifier.summary.emailTemplate=C:/Netscape/Server4/cert-testCA/emails/renewJobSummary.bt
jobsScheduler.job.certRenewalNotifier.summary.enabled=true
jobsScheduler.job.certRenewalNotifier.summary.itemTemplate=C:/Netscape/Server4/cert-testCA/emails/renewJobItem.bt
jobsScheduler.job.certRenewalNotifier.summary.recipientEmail=ca_agent1@netscape.com,ca_agent2@netscape.com
jobsScheduler.job.certRenewalNotifier.summary.senderEmail=CAAdmin@netscape.com

jobsScheduler.job.requestInQueueNotifier.cron=0 0 * * 0
jobsScheduler.job.requestInQueueNotifier.enabled=false
jobsScheduler.job.requestInQueueNotifier.pluginName=RequestInQueueJob
jobsScheduler.job.requestInQueueNotifier.subsystemId=ca
jobsScheduler.job.requestInQueueNotifier.summary.emailSubject=Requests in Queue Summary Report
jobsScheduler.job.requestInQueueNotifier.summary.emailTemplate=C:/Netscape/Server4/cert-testCA/emails/reqInQSummary.html
jobsScheduler.job.requestInQueueNotifier.summary.enabled=true
jobsScheduler.job.requestInQueueNotifier.summary.recipientEmail=ca_agent1@netscape.com,ca_agent2@netscape.com
jobsScheduler.job.requestInQueueNotifier.summary.senderEmail=CAAdmin@netscape.com

jobsScheduler.job.unpublishExpiredCerts.cron=0 0 * * 6
jobsScheduler.job.unpublishExpiredCerts.enabled=true
jobsScheduler.job.unpublishExpiredCerts.pluginName=UnpublishExpiredJob
jobsScheduler.job.unpublishExpiredCerts.summary.emailSubject=Expired Certificate Removal Job Summary
jobsScheduler.job.unpublishExpiredCerts.summary.emailTemplate=C:/Netscape/Server4/cert-testCA/emails/unpublishCertsJob1.htm
jobsScheduler.job.unpublishExpiredCerts.summary.enabled=true
jobsScheduler.job.unpublishExpiredCerts.summary.itemTemplate=C:/Netscape/Server4/cert-testCA/emails/unpublishCertsItem.html
jobsScheduler.job.unpublishExpiredCerts.summary.recipientEmail=ca_agent1@netscape.com,ca_agent2@netscape.com
jobsScheduler.job.unpublishExpiredCerts.summary.senderEmail=CAAdmin@netscape.com

```

To change the configuration by editing the configuration file, follow the instructions in “Changing the Configuration by Editing the Configuration File” on page 72.

## Job Plug-in Implementation and Instance

Jobs are implemented as Java classes, which are then registered with Certificate Management System as plug-ins. You can use a given implementation of a job plug-in module and configure multiple instances of it. Each instance must have a unique name (an alphanumeric string with no spaces) and can contain different input parameter values to apply to different jobs. In other words, a given job implementation can be shared by multiple configurations. You can also distinguish the applicability of configured instances by including appropriate instance names.

## Managing Jobs

This section explains how to use the CMS window to perform the following operations:

- Adding a Job
- Deleting a Job
- Modifying a Job
- Setting the Job Scheduler Frequency

For information on adding or changing job-specific information in the configuration file, see “Job Scheduler Parameters in the Configuration File” on page 385.

### Adding a Job

Adding a job to the CMS configuration involves creating a new instance of an already registered plug-in module, assigning a unique name (an alphanumeric string with no spaces) for the instance, and entering appropriate values for the parameters that define the plug-in module you want to create an instance of.

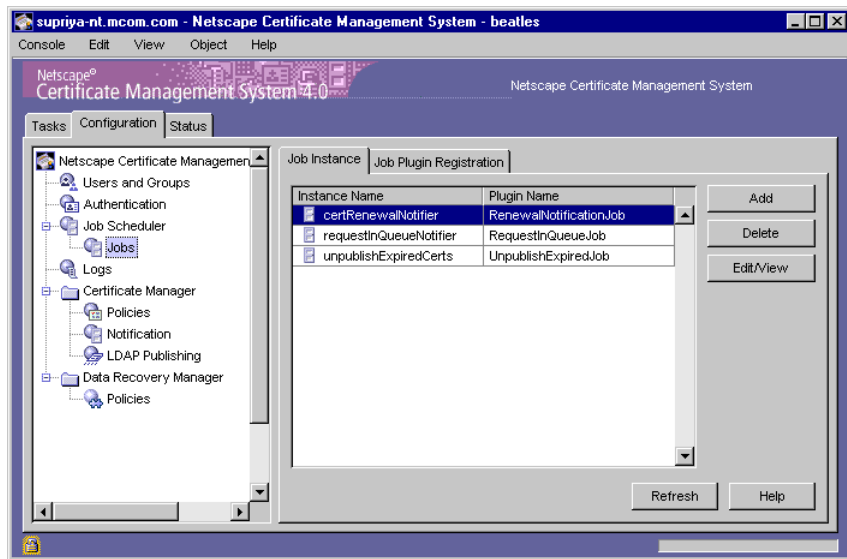
When you add a job, the CMS configuration is updated with the appropriate information.

**Note** Certificate Management System executes this job only if the Job Scheduler is turned on or enabled. For information on scheduling the interval for executing the job, see “Setting the Job Scheduler Frequency” on page 394.

To add a job to the CMS configuration:

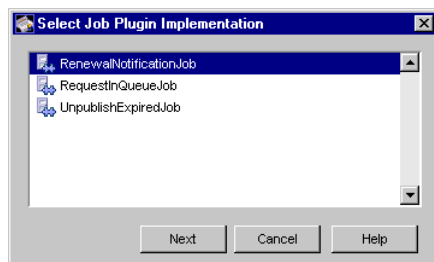
1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, click Job Scheduler, then click Jobs.

The Job Instance tab appears. It lists any currently configured jobs. For information about this tab, see “Job Instance Tab” on page 383.



4. Click Add.

The Select Job Plugin Implementation window appears. It lists the currently registered job plug-in modules.



5. Select a plug-in module.

The following choices are the ones provided out of the box with Certificate Management System. If you have registered any custom job plug-in modules, they too will be available for selection.

- **RenewalNotificationJob**

A schedulable job that notifies end entities by email that their certificates are about to expire and must be renewed. It can also send a summary of these notices to agents. For details, see “Certificate Renewal Notifications” on page 347.

- **RequestInQueueJob**

A schedulable job that notifies agents at regular intervals of the current state of the request queue. For details, see “Notification of Request Queue Status” on page 353.

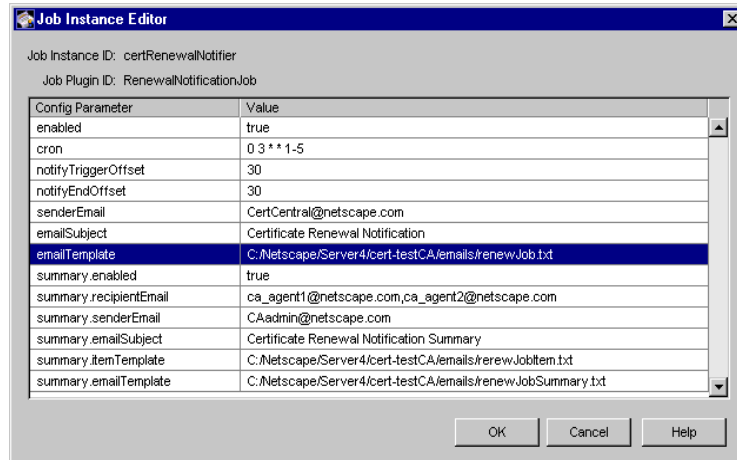
- **UnpublishExpiredJob**

A schedulable job that updates the configured publishing directory periodically by removing expired certificates. It can also send a summary of removed certificates to agents or administrators. For details, see “Directory Update and Notification” on page 358.

For the purposes of this instruction, assume that you selected `RenewalNotificationJob`.

6. Click Next.

The Configure Job Instance Parameters window appears. It lists the configuration information required for this job. For information on how this window works, see “Job Instance Editor” on page 384.



7. In the Job Instance ID field, type a unique name for this job that will help you identify it.

For the name, be sure to use an alphanumeric string with no spaces.

8. In the configuration area, specify the required information by filling in parameter values in the text fields in the right column.

If you do not want to set any restrictions on a particular parameter, leave its value field blank.

9. Click OK.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

## Deleting a Job

You can delete unwanted jobs from the CMS configuration, by using the CMS window. If you think you might need a job in the future, instead of deleting it from the configuration you should disable it by setting the `enable` parameter value to `false`. In this way, you can avoid re-creating the job in the future. Because Certificate Management System executes only those jobs that are currently enabled, keeping unwanted jobs in a disabled state in the configuration does not affect the server's functioning.

To delete a job from the CMS configuration:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, click Job Scheduler, then click Jobs.

The Job Instance tab appears. It lists any currently configured jobs. For information about this tab, see “Job Instance Tab” on page 383.

4. In the Instance Name list, select the job you want to delete and click Delete.
5. When prompted, confirm the delete action.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

## Modifying a Job

Modifying a job involves changing the configuration parameter values of the job instance; you cannot change the name of a job. To change the name of a job, create a new job using the same job plug-in module (that you used to create the job you want to rename) with the same parameter values, and delete the old one.

**Note** Certificate Management System executes jobs only if the Job Scheduler is turned on or enabled. For information on scheduling the interval for executing the job, see “Setting the Job Scheduler Frequency” on page 394.

During installation, both Certificate Manager and Registration Manager create default jobs. Table 14.1 lists these jobs. After installation, you must verify whether you want to use these jobs, check how these jobs are configured, and make the appropriate configuration changes. If you don't want to use a job, delete it from the configuration following the instructions in “Deleting a Job” on page 391. If you want to create a new job, follow the instructions in “Adding a Job” on page 387.

**Table 14.1** Default jobs created for a Certificate Manager or Registration Manager

Job instance name	Job plug-in module name
<code>certRenewalNotifier</code>	<code>RenewalNotificationJob</code> See “Certificate Renewal Notifications” on page 347.
<code>requestInQueueNotifier</code>	<code>RequestInQueueJob</code> See “Directory Update and Notification” on page 358.
<code>unpublishExpiredCerts</code>	<code>UnpublishExpiredJob</code> See “Directory Update and Notification” on page 358.

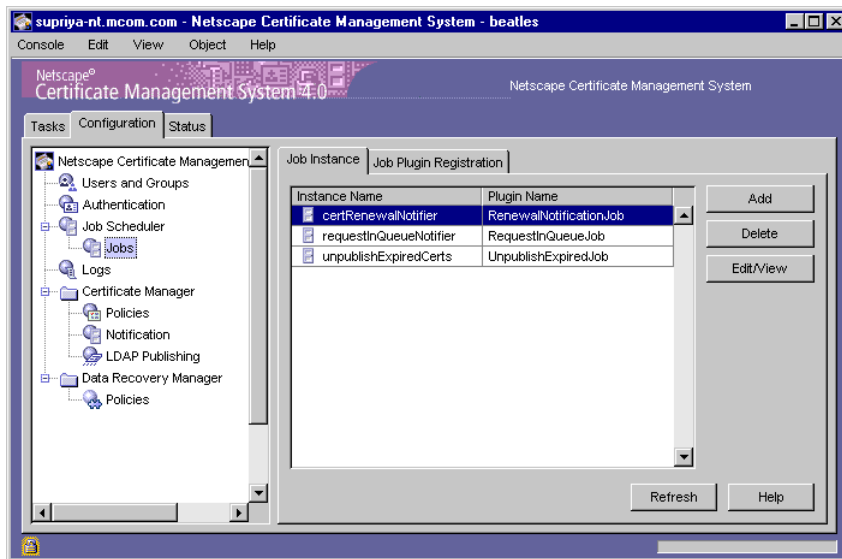
To modify a configured job in the CMS configuration:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.



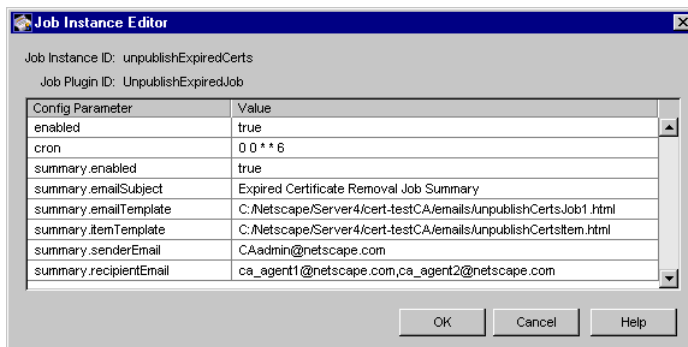
3. In the navigation tree, click Job Scheduler, then click Jobs.

The Job Instance tab appears. It lists any currently configured jobs. For information about this tab, see “Job Instance Tab” on page 383.



4. In the Instance Name list, select the job you want to modify and click Edit.

The Configure Job Instance Parameters window appears, showing how this job is currently configured. For information on how this window works, see “Job Instance Editor” on page 384.



5. Make the necessary changes by filling in parameter values in the text fields in the right column.

If you do not want to set any restrictions on a particular parameter, leave its value field blank.

6. Click OK.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

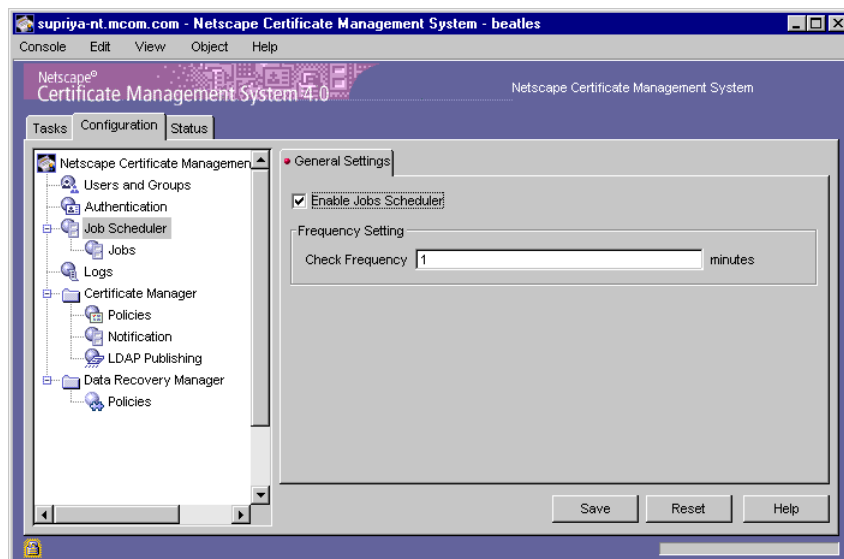
## Setting the Job Scheduler Frequency

You can schedule the frequency at which the Job Scheduler daemon should check if any of the configured jobs need to be executed. To do this:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.

3. In the navigation tree, click Job Scheduler.

The General Settings tab appears. It shows whether the Job Scheduler component is currently enabled or disabled.



4. Enter information as appropriate:

**Enable Job Scheduler.** Check this option to enable the Job Scheduler. To disable the Job Scheduler uncheck the option; disabling turns off all the jobs.

**Check Frequency.** Type the frequency at which the Job Scheduler daemon thread should wake up and call the configured jobs that meet the cron specification (see “Schedule for Executing Jobs” on page 363). By default, it is set to one minute.

5. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

# Managing Job Plug-in Modules

This section explains how to use the CMS window to perform the following operations:

- Registering a Job Plug-in Module
- Deleting a Job Scheduler Plug-in

For information on adding or changing job-specific information in the configuration file, see “Job Scheduler Parameters in the Configuration File” on page 385.

## Registering a Job Plug-in Module

You can register custom job plug-in modules from the CMS window. Before registering a module, be sure to put the Java class for the plug-in module in the `classes` directory.

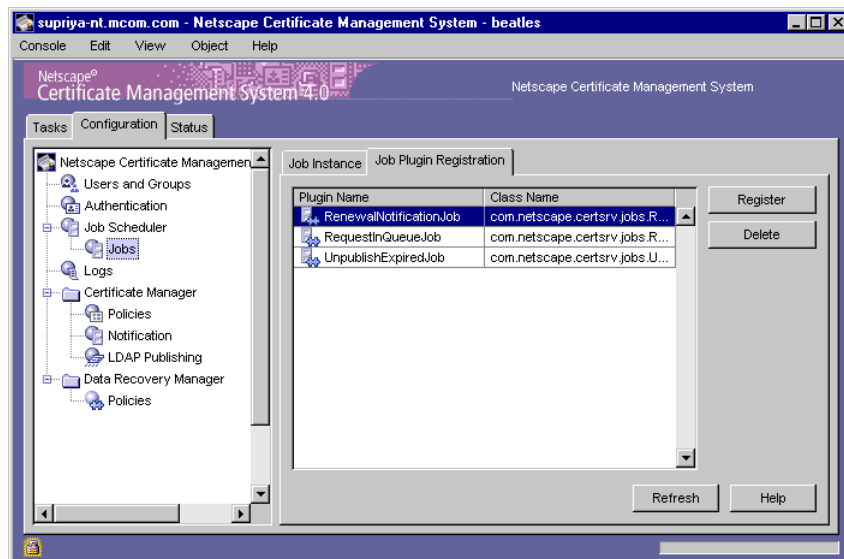
To register a job plug-in module in the CMS framework:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, click Job Scheduler, then click Jobs.

The Job Instance tab appears. It lists any currently configured jobs.

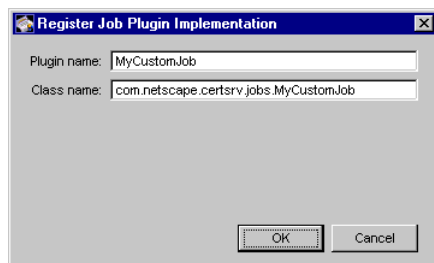
4. Click the Job Plugin Registration tab.

The Job Plugin Registration tab appears. It lists currently registered job plug-in modules. For information about this tab, see “Job Plugin Registration Tab” on page 384.



5. Click Register.

The Register Job Scheduler Plugin Implementation window appears. For information on how this window works, see “Job Plugin Registration Editor” on page 385.



6. Specify information as appropriate:

**Plugin name.** Type the name of the plug-in module.

**Class name.** Type the full name of the class for this plug-in module—that is, the path to the implementing Java class. If this class is part of a package, be sure to include the package name. For example, if you are registering a class named `myJob` and if this class is in a package named `com.myCompany`, type `com.myCompany.myJob`.

7. Click OK.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

## Deleting a Job Scheduler Plug-in

You can delete unwanted job plug-in modules by using the CMS window. Before deleting a module, be sure to delete all the instances that are based on this module; for instructions, see “Deleting a Job” on page 391.

To delete a job plug-in from the CMS framework:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, click Job Scheduler, then click Jobs.

The Job Instance tab appears. It lists any currently configured instances.

4. Click the Job Plugin Registration tab.

The Job Plugin Registration tab appears. It lists currently registered job plug-in modules. For information about this tab, see “Job Plugin Registration Tab” on page 384.

5. In the Plugin Name list, select the plug-in module you want to delete and click Delete.
6. When prompted, confirm the delete action.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

# 6

## *Policies*

*Chapter 15* Introduction to Policy

*Chapter 16* Configuring Policies





# Introduction to Policy

You can configure the main subsystems of Netscape Certificate Management System (CMS)—the Certificate Manager, Registration Manager, and Data Recovery Manager—to apply certain organizational policies on an end entity's certificate enrollment and management requests before servicing them. For example, some of the policies you might want a Certificate Manager to impose on these requests may include setting a minimum and maximum limit on validity period and key length of certificates, setting extensions based on the end entity's role within an organization, setting signing algorithms, and so on.

This chapter provides an overview of policy in general, and it explains the various policy plug-ins supported out of the box.

The chapter has the following sections:

- What Is Policy? (page 402)
- Policy Rules (page 403)
- Policy Processor (page 408)
- Built-in Policy Plug-in Modules (page 409)

# What Is Policy?

*Policy* refers to a set of rules that Certificate Management System uses to evaluate or verify an incoming request from an end entity and to determine the outcome; the incoming requests that are governed by policies include certificate issuance, certificate renewal, certificate revocation, key archival, and key recovery requests. For example, in the case of a certificate issuance request, the outcome would be the certificate content.

- A Certificate Manager's policy can include rules for evaluating certificate formulation, signing, renewal, and revocation requests. For example, you can configure a Certificate Manager's policy to impose restrictions on validity length, key type, key length, subject name, extensions, and signing algorithm during certificate issuance.
- A Registration Manager's policy can include rules for verifying incoming certificate issuance, renewal, and revocation requests from end entities in order to formulate the certificate content before forwarding the requests to a Certificate Manager for signing. For example, you can configure a Registration Manager's policy to impose restrictions on validity period, key length, subject name, and extensions. In general, policies for Registration Manager are largely the same as for Certificate Manager.
- A Data Recovery Manager's policy can include rules for verifying users' encryption private key archival and recovery requests.

Using policies, you can configure Certificate Management System to perform one or more of the following operations on each certificate issuance or management request it receives:

- Screen the request for specific content, and modify, reject, or defer (for agent approval) it accordingly. For example, the request might be checked for the inclusion of organizational constraints, such as key algorithm, key size, validity period, or a particular signing algorithm; if it did not meet the requirement, the subsystem would modify the request or return an error, depending on the severity of the problem.
- Set common attributes, such as extensions for user and server certificate requests.

# Policy Rules

A policy rule refers to a uniquely configured instance of any policy plug-in implementation; see “Built-in Policy Plug-in Modules” on page 409 for a list of plug-in modules provided for Certificate Management System. For example, you can use the plug-in module provided for setting validity periods on certificates to configure a policy rule that forces validity periods for all client certificates issued by a Certificate Manager to fall within a predetermined range, say between 6 and 24 months. A subsystem’s policy configuration can consist of one or more policy rules, each performing one or more of the following operations:

- Validate the request content by comparing it with configured criteria; reject, modify, or defer (for agent approval) the request if any of the request parameters are invalid.
- Build certificate content—for example, set common extensions and the validity period.
- Enforce organizational constraints, such as subject name, key algorithm, key size, and validity period.
- Determine whether the private key should be archived.

Keep in mind that the server applies the rules when processing end-entity requests and after agent approval (for deferred requests).

## Types of Policy Rules

Certificate Management System supports distinct policy rules for each of the operations that end entities perform—certificate enrollment, renewal, and revocation, and key archival and recovery. Consequently, there are five broad categories of policies, corresponding to these types of operations:

- Enrollment policies
- Renewal policies
- Revocation policies
- Key archival policies

- Key recovery policies

To facilitate this classification, Certificate Management System supports a parent interface for a generic policy rule and other operation-specific interfaces that extend the parent interface. Check the SDK directory on the product CD. You can also download the CMS SDK from this site:

<http://home.netscape.com/eng/server/cms>

## Using Predicates in Policy Rules

You can use predicates in a policy rule. A predicate indicates whether the rule that contains the predicate applies to a request. If you specify a predicate as part of the rule configuration, the policy rule applies that predicate based on request attributes to determine whether the rule is applicable for a request.

The policy predicate is an expression. You form the expression using standard variables that return strings, integers, or enumeration of strings/integers, and relational operators (AND or OR). For details, see “Expression Support for Predicates” on page 404 and “Attributes for Predicates” on page 406.

## Expression Support for Predicates

You form an expression using an attribute, its value, and one or more of the operators listed in Table 15.1. For a list of attributes, see “Attributes for Predicates” on page 406.

Table 15.1 Predicates in policy: supported comparison and logical operators

Operator	Description
==	Equal to
!=	Not equal to
AND	Logical operator AND
OR	Logical operator OR

The expression parsing support currently supports only two comparison operators (==, !=) and two relational operators (AND, OR).

Policy expressions are formed with the following rules:

`PrimitiveExpression` | `AndExpression` | `OrExpression`

- `PrimitiveExpression` is equal to: `Attribute op Value`  
 where  
`Attribute` can be a string  
`op` can be any of these operators: `==` or `!=`  
`Value` can be a string
- `AndExpression` is equal to: `Expression AND Expression`
- `OrExpression` is equal to: `Expression OR Expression`

In an expression, the AND operator takes precedence over an OR operator. For example, the expression

```
certType == client AND ou==Engineering OR certType == ca
```

is interpreted as

```
(certType == client AND ou==Engineering) OR certType == ca
```

Certificate Management System evaluates an expression based on the attributes in the request. The attributes are filled in by servlets from the HTTP input forms used for request submission. Some attributes, such as passwords typed in the form are not stored in the request. Other attributes regarding the end entity are set on the request after successful authentication. The servlets also interpret the form content, for example, retrieving the key material out of the `KEYGEN` or `PKCS #10` information and setting the key in the certificate content. They can also set additional attributes related to the certificate content on the request. In general, you can configure which attributes—for example, sensitive attributes such as passwords—should or shouldn't be stored in the request.

All data related to an end entity is gathered at the servlet level and set on the request before the request is passed to the policy subsystem. The policy subsystem applies configured policy rules on the request, determines whether the request needs agent approval, performs constraint- and extension-specific checks on the request attributes, and then formulates the certificate content by adding the appropriate information, such as the validity period and extensions.

The expression queries the request for the attributes, compares the value returned with the value provided in the predicate, and returns a boolean result.

The following are sample predicates:

```
certType == client AND ou == Engineering
```

```
certType == server AND o == Netscape OR certType == ca
```

## Attributes for Predicates

Attributes for predicates can come from any of the following:

- Input form—that is, the HTML form that end entities use for submitting certificate requests.
- Authentication token—what the authentication subsystem returns after successfully authenticating an end entity.
- A service—for example, a Certificate Manager, Registration Manager, or Data Recovery Manager service can add certain attributes to the end-entity request.
- Policy processor—what the policy subsystem returns after subjecting the end-entity request to policy checking.

Table 15.2 lists default attributes that are supported by various request object implementations.

Table 15.2 Attributes supported by request object implementations

Request type	Variable name	Description	Object type
<b>Default attributes from an input form:</b>			
Enrollment	certType	<p>The certificate type. Possible values include the following:</p> <ul style="list-style-type: none"> <li>• <code>client</code> (client certificates)</li> <li>• <code>server</code> (server certificate)</li> <li>• <code>Cisco-router</code> (router certificate)</li> <li>• <code>ra</code> (Registration Manager's signing certificate)</li> <li>• <code>ca</code> (Certificate Manager's CA signing certificate)</li> <li>• <code>other</code></li> </ul>	String
Enrollment	requestFormat	<p>The certificate request format. Possible values include the following:</p> <ul style="list-style-type: none"> <li>• <code>keygen</code></li> <li>• <code>pkcs10</code></li> </ul>	String
Renewal	requestFormat	<p>The certificate request format. Possible values include the following:</p> <ul style="list-style-type: none"> <li>• <code>clientAuth</code></li> <li>• <code>pkcs10</code></li> </ul>	String
<b>Default attributes from an authentication token:</b>			
(Upon successful authentication these attributes go into an enrollment request)			
Enrollment	authMgrImplName	The name of the authentication plug-in module that authenticated the request.	String
Enrollment	authMgrInstName	The name of the authentication instance that authenticated the request.	String

# Policy Processor

Each subsystem—the Certificate Manager, Registration Manager, or Data Recovery Manager—has its own policy processor. Each processor subjects an incoming request to the applicable policy rules for that subsystem.

When a subsystem starts up, its policy processor reads the current policy configurations from the configuration file, initializes them, and classifies them based on their type (see “Types of Policy Rules” on page 403). Then, when the subsystem receives an authenticated request, its request processor invokes the policy processor to apply policies on that request. The policy processor applies the rules on the request based on the request type. The policy processor also filters the rules based on predicates (see “Using Predicates in Policy Rules” on page 404).

Note that the policy processor applies only the *enabled* policy rules, in the order in which they are configured, before determining the final outcome. Each rule the processor executes returns a `PolicyResult` object. Three return values are possible:

- `PolicyResult.REJECTED` (indicates that the request failed the rule)
- `PolicyResult.DEFERRED` (indicates that the request requires agent approval)
- `PolicyResult.ACCEPTED` (indicates that the request passed the rule)

After all the policy rules are applied, the processor determines the status of the request (in this order):

1. If the request failed any policy rule (that is, if any of the policy rules returned a `PolicyResult.REJECTED` value), the processor rejects the request. The rule that rejected the request sets appropriate error messages on the request.
2. If at least one of the policy rules requires agent approval for the request (that is, if any of the policy rules returned a `PolicyResult.DEFERRED` value), the processor stores the request in the request queue for agent approval.



3. If the request passes all the policy rules (that is, all policy rules returned a `PolicyResult.ACCEPTED` value), the request gets serviced—for example the certificate is issued or renewed.

## Built-in Policy Plug-in Modules

Certificate Management System comes with various policy plug-in modules that help you govern its certificate generation and management operations.

The modules provided out of the box can be categorized into two groups:

- Constraints-Specific Policy Plug-in Modules
- Extension-Specific Policy Plug-in Modules

Note that Certificate Management System doesn't come with key archival and recovery policy modules.

## Constraints-Specific Policy Plug-in Modules

Constraints-specific policy plug-in modules help you define rules or constraints that Certificate Management System must use to evaluate an incoming request. Table 15.3 lists the constraints-specific policy plug-in modules provided out of the box. You can use these modules, as applicable, for defining the required policy rules for your Certificate Manager or Registration Manager; no plug-ins are provided for the Data Recovery Manager.

Table 15.3 Constraints-specific policy plug-in modules provided out of the box

Plug-in module name	Description
DefaultRevocation	<p>Use this to configure the server to revoke only those certificates that are currently valid; expired certificates cannot be revoked. For more information, see “Default Revocation Policy” on page 411.</p> <p>Both the Certificate Manager and Registration Manager provide this plug-in module.</p>
DSAKeyConstraints	<p>Use this to configure the server to certify only those DSA keys that have specific key lengths. For more information, see “DSA Key Constraints Policy” on page 413.</p> <p>Both the Certificate Manager and Registration Manager provide this plug-in module.</p>
KeyAlgorithmConstraints	<p>Use this to configure the server to certify only those keys that are generated using one of the permitted algorithms, such as RSA or DSA. For more information, see “Key Algorithm Constraints Policy” on page 416.</p> <p>Both the Certificate Manager and Registration Manager provide this plug-in module.</p>
RenewalValidityConstraints	<p>Use this to configure the server to enforce the number of days before which a currently active certificate can be renewed and to set a new validity period for the renewed certificate. For more information, see “Renewal Validity Constraints Policy” on page 419.</p> <p>Both the Certificate Manager and Registration Manager provide this plug-in module.</p>

Table 15.3 Constraints-specific policy plug-in modules provided out of the box (Continued)

Plug-in module name	Description
<code>RSAPublicKeyConstraints</code>	<p>Use this to configure the server to certify only those RSA keys that have specific key lengths. For more information, see “RSA Key Constraints Policy” on page 422.</p> <p>Both the Certificate Manager and Registration Manager provide this plug-in module.</p>
<code>ValidityConstraints</code>	<p>Use this to configure the server to check whether the validity period of a certificate falls within a specific validity period. For more information, see “Validity Constraints Policy” on page 426.</p> <p>Both the Certificate Manager and Registration Manager provide this plug-in module.</p>

## Default Revocation Policy

The default revocation policy applies to end-entity certificate revocation. The policy enforces a rule that only currently valid certificates can be revoked; an expired certificate cannot be revoked. The policy also looks at the *revocation reason* specified in the revocation form and sets the appropriate `CRLReason` extension.

### Plug-in for Default Revocation Policy

The plug-in module provided for the default revocation policy is identified as follows:

`com.netscape.certsrv.policy.DefaultRevocation`

- In the configuration file, the implementation for this module is identified as follows; you can find it in the Policy section:

```
<subsystem>.Policy.impl.DefaultRevocation.class=com.netscape
.certsrv.policy.DefaultRevocation
```

where `<subsystem>` indicates `ca` or `ra` (prefix identifying the subsystem)

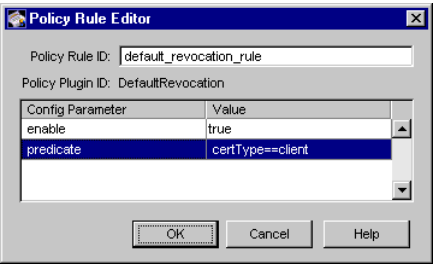
- In the CMS window, the implementation for this module is identified as follows; you can find it in the Policy Plugin Registration tab of the corresponding subsystem:

`DefaultRevocation`

### Configurable Parameters

Figure 15.1 shows how the configurable parameters pertaining to the `DefaultRevocation` policy plug-in implementation are displayed in the CMS window.

Figure 15.1 Parameters and values for the `DefaultRevocation` module



The configuration shown in Figure 15.1 enforces a rule that only valid client certificates can be revoked; expired certificates cannot be revoked.

Table 15.4 gives details about each of these parameters.

Table 15.4 Configurable parameters for the default revocation policy and their values

Parameter name	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. To enable the rule, enter <code>true</code>; to disable the rule, enter <code>false</code>. By default, the rule is enabled.</p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li>• <code>true</code></li> <li>• <code>false</code></li> </ul> <p>Object type: String</p>
<code>predicate</code>	<p>Specifies the predicate expression for this rule. If you want the rule to be applied to all certificate requests, leave the field blank. By default, the rule is applied to all certificate requests.</p> <p>Example: <code>certType==client</code></p> <p>Permissible values: See “Using Predicates in Policy Rules” on page 404.</p> <p>Object type: String</p>

## DSA Key Constraints Policy

The DSA key constraints policy applies to end-entity certificate enrollment and renewal requests. The policy imposes constraints on the following:

- The minimum and maximum sizes for keys
- The exponent sizes

The policy restricts the key size to one of the sizes, such as 512, 1024, or 2048, supported by the Certificate Management System. In other words, this policy allows you to set up restrictions on the lengths of public keys certified by Certificate Management System. For example, you can configure a Certificate Manager to certify public keys up to 1024 bits in length for end users.

## Plug-in for DSA Key Constraints Policy

The plug-in module provided for the DSA key constraints policy is identified as follows:

```
com.netscape.certsrv.policy.DSAKeyConstraints
```

- In the configuration file, the implementation for this module is identified as follows; you can find it in the Policy section:

```
<subsystem>.Policy.impl.DSAKeyConstraints.class=com.netscape
.certsrv.policy.DSAKeyConstraints
```

where <subsystem> indicates ca or ra (prefix identifying the subsystem)

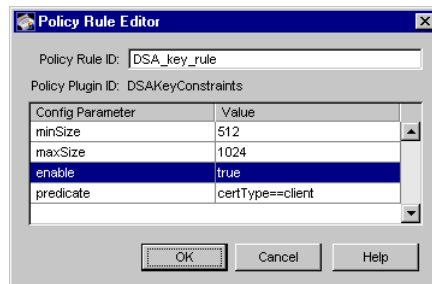
- In the CMS window, the implementation for this module is identified as follows; you can find it in the Policy Plugin Registration tab of the corresponding subsystem:

```
DSAKeyConstraints
```

## Configurable Parameters

Figure 15.2 shows how the configurable parameters pertaining to the DSAKeyConstraints policy plug-in implementation are displayed in the CMS window.

Figure 15.2 Parameters and values for the DSAKeyConstraints module



The configuration shown in Figure 15.2 restricts the minimum and maximum key sizes for all client certificates to 512 and 1024, respectively.

Table 15.5 gives details about each of these parameters.

Table 15.5 Configurable parameters for the DSA key constraints policy and their values

Parameter name	Description
<code>minSize</code>	<p>Specifies the minimum length for the key (the length of the modulus in bits). Enter the length in bits. The value must be smaller than or equal to the one specified by the <code>maxSize</code> parameter. The default value is 512.</p> <p>In general, a longer key size results in a key pair that is more difficult to crack. You may want to enforce a minimum length to ensure a minimum level of security.</p> <p>Example: 512</p> <p>Permissible values: Any of the following:</p> <ul style="list-style-type: none"> <li>• 512</li> <li>• 1024</li> <li>• 2048</li> </ul> <p>Object type: Integer</p>
<code>maxSize</code>	<p>Specifies the maximum length for the key. Enter the length in bits. The default value is 2048.</p> <p>The current export restrictions imposed by the United States limit the exported versions of Netscape software to supporting public keys no longer than 512 bits. To allow keys certified by your server to work with software across national borders, you may want to enforce a maximum length of 512 bits.</p> <p>Example: 1024</p> <p>Permissible values: Any of the following:</p> <ul style="list-style-type: none"> <li>• 512</li> <li>• 1024</li> <li>• 2048</li> </ul> <p>Object type: Integer</p>

Table 15.5 Configurable parameters for the DSA key constraints policy and their values (Continued)

Parameter name	Description
enable	<p>Specifies whether the rule is enabled or disabled. To enable the rule, enter <code>true</code>; to disable the rule, enter <code>false</code>. By default, the rule is enabled.</p> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"><li>• <code>true</code></li><li>• <code>false</code></li></ul> <p>Object type: String</p>
predicate	<p>Specifies the predicate expression for this rule. If you want the rule to be applied to all certificate requests, leave the field blank. By default, the rule is applied to all certificate requests.</p> <p>Example: <code>certType==client</code></p> <p>Permissible values: See “Using Predicates in Policy Rules” on page 404.</p> <p>Object type: String</p>

## Key Algorithm Constraints Policy

The key algorithm constraints policy applies to end-entity certificate enrollment and renewal requests. The policy restricts the key algorithm requested to one of the algorithms, such as RSA or DSA, supported by Certificate Management System. In other words, this policy allows you to set restrictions on the types of public keys certified by Certificate Management System.

For example, you can configure a Certificate Manager to certify only those public keys that comply with the PKCS-1 RSA Encryption Standard.



## Plug-in for Key Algorithm Constraints Policy

The plug-in module provided for the key algorithm constraints policy is identified as follows:

```
com.netscape.certsrv.policy.KeyAlgorithmConstraints
```

- In the configuration file, the implementation for this module is identified as follows; you can find it in the Policy section:

```
<subsystem>.Policy.impl.KeyAlgorithmConstraints.class=com.netscape.certsrv.policy.KeyAlgorithmConstraints
```

where <subsystem> indicates ca or ra (prefix identifying the subsystem)

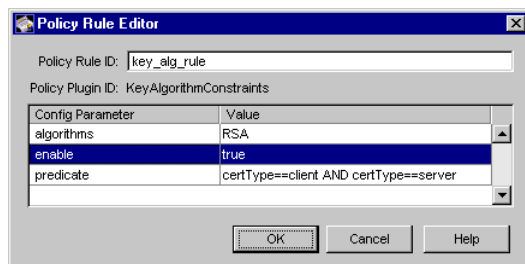
- In the CMS window, the implementation for this module is identified as follows; you can find it in the Policy Plugin Registration tab of the corresponding subsystem:

```
KeyAlgorithmConstraints
```

## Configurable Parameters

Figure 15.3 shows how the configurable parameters pertaining to the `KeyAlgorithmConstraints` policy plug-in implementation are displayed in the CMS window.

Figure 15.3 Parameters and values for the `KeyAlgorithmConstraints` module



The configuration shown in Figure 15.3 restricts the key algorithm of all client and server certificates to RSA.

Table 15.6 gives details about each of these parameters.

Table 15.6 Configurable parameters for the key algorithm constraints policy and their values

Parameter name	Description
<code>algorithms</code>	<p>Specifies the key type the server should certify. The default is <code>RSA</code>.</p> <p>Example: <code>RSA</code></p> <p>Permissible values: Either or both of the following; if entering both, separate them by a comma:</p> <ul style="list-style-type: none"> <li>• <code>RSA</code></li> <li>• <code>DSA</code></li> </ul> <p>Object type: String</p>
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. To enable the rule, enter <code>true</code>; to disable the rule, enter <code>false</code>. By default, the rule is enabled.</p> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li>• <code>true</code></li> <li>• <code>false</code></li> </ul> <p>Object type: String</p>
<code>predicate</code>	<p>Specifies the predicate expression for this rule. If you want the rule to be applied to all certificate requests, leave the field blank. By default, the rule is applied to all certificate requests.</p> <p>Example: <code>certType==client AND certType==server</code></p> <p>Permissible values: See “Using Predicates in Policy Rules” on page 404.</p> <p>Object type: String</p>

## Renewal Validity Constraints Policy

The renewal validity constraints policy applies to end-entity certificate renewals. It governs the formulation of content in the renewed certificate based on the currently issued certificate. The policy enforces the following rules:

- The number of days before expiration that a currently active certificate can be renewed.
- The new validity period for the renewed certificate, starting from the end of the period in the old certificate.

The renewal process to which this policy is applied can be manual (request needs to be approved by an agent) or automated; for details, see “Certificate Renewal” on page 619. In both cases, the currently issued certificate must be either presented during SSL client authentication by the end entity or selected by the agent approving the renewal request.

You should consider using this policy if you want to enforce the following:

- The number of days before expiration that end entities can renew their currently active or valid certificates. For example, you may want to enforce a policy that prevents end entities from renewing their certificates any earlier than 15 days before expiration.
- The validity period of the renewed certificate. For example, you may want to enforce a policy that sets the validity period of renewed certificates to a minimum of 180 days; the renewal period starts from the ending period in the certificate presented for renewal.

### Plug-in for Renewal Validity Constraints Policy

The plug-in module provided for the renewal validity constraints policy is identified as follows:

`com.netscape.certsrv.policy.RenewalValidityConstraints`

- In the configuration file, the implementation for this module is identified as follows; you can find it in the Policy section:

```
<subsystem>.Policy.impl.RenewalValidityConstraints.class=com
.netscape.certsrv.policy.RenewalValidityConstraints
```

where `<subsystem>` indicates `ca` or `ra` (prefix identifying the subsystem)

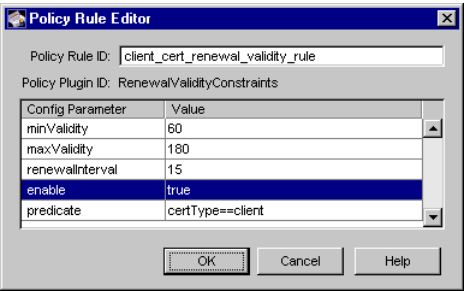
- In the CMS window, the implementation for this module is identified as follows; you can find it in the Policy Plugin Registration tab of the corresponding subsystem:

`RenewalValidityConstraints`

### Configurable Parameters

Figure 15.4 shows how the configurable parameters pertaining to the `RenewalValidityConstraints` policy plug-in implementation are displayed in the CMS window.

Figure 15.4 Parameters and values for the `RenewalValidityConstraints` module



The configuration shown in Figure 15.4 enforces a rule that only those client certificates that are due to expire within the next 15 days can be renewed. The renewed certificates are valid for at least 60 days (two months) and require renewing after 180 days (six months).

Table 15.7 gives details about each of these parameters.

Table 15.7 Configurable parameters for the renewal validity constraints policy and their values

Parameter name	Description
<code>minValidity</code>	<p>Specifies the minimum validity period, in days, for renewed certificates. The default value is 180 days.</p> <p>Example: 60</p> <p>Permissible values: As applicable</p> <p>Object type: Integer</p>
<code>maxValidity</code>	<p>Specifies the maximum validity period, in days, for renewed certificates. The default value is 730 days.</p> <p>Example: 180</p> <p>Permissible values: As applicable</p> <p>Object type: Integer</p>
<code>renewalInterval</code>	<p>Specifies how many days before its expiration that a certificate can be renewed. The default value is 15 days.</p> <p>Example: 15</p> <p>Permissible values: As applicable</p> <p>Object type: Integer</p>
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. To enable the rule, enter <code>true</code>; to disable the rule, enter <code>false</code>. By default, the rule is enabled.</p> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li>• <code>true</code></li> <li>• <code>false</code></li> </ul> <p>Object type: String</p>

Table 15.7 Configurable parameters for the renewal validity constraints policy and their values (Continued)

Parameter name	Description
predicate	<p>Specifies the predicate expression for this rule. If you want the rule to be applied to all certificate requests, leave the field blank. By default, the rule is applied to all certificate requests.</p> <p>Example: <code>certType==client</code></p> <p>Permissible values: See “Using Predicates in Policy Rules” on page 404.</p> <p>Object type: String</p>

## RSA Key Constraints Policy

The RSA key constraints policy applies to end-entity certificate enrollment and renewal requests. The policy imposes constraints on the following:

- The minimum and maximum sizes for keys
- The exponent sizes

The policy restricts the key size to one of the sizes, such as 512, 1024, or 2048, supported by Certificate Management System. In other words, this policy allows you to set restrictions on the length of public keys certified by the Certificate Management System.

For example, you can configure a Certificate Manager to certify public keys up to 1024 bits in length for end users.

## Plug-in for RSA Key Constraints Policy

The plug-in module provided for the RSA key constraints policy is identified as follows:

```
com.netscape.certsrv.policy.RSAKeyConstraints
```

- In the configuration file, the implementation for this module is identified as follows; you can find it in the Policy section:

```
<subsystem>.Policy.impl.RSAKeyConstraints.class=com.netscape
.certsrv.policy.RSAKeyConstraints
```

where <subsystem> indicates ca or ra (prefix identifying the subsystem)

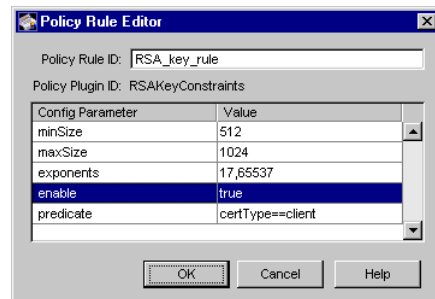
- In the CMS window, the implementation for this module is identified as follows; you can find it in the Policy Plugin Registration tab of the corresponding subsystem:

```
RSAKeyConstraints
```

## Configurable Parameters

Figure 15.5 shows how the configurable parameters pertaining to the RSAKeyConstraints policy plug-in implementation are displayed in the CMS window.

Figure 15.5 Parameters and values for the RSAKeyConstraints module



The configuration shown in Figure 15.5 restricts the minimum and maximum key sizes for all client certificates to 512 and 1024, respectively.

Table 15.8 gives details about each of these parameters.

Table 15.8 Configurable parameters for the RSA key constraints policy and their values

Parameter name	Description
<code>minSize</code>	<p>Specifies the minimum length for the key (the length of the modulus in bits). Enter the length in bits. The value must be smaller than or equal to the one specified by the <code>maxSize</code> parameter. The default value is 512.</p> <p>In general, a longer key size results in a key pair that is more difficult to crack. You may want to allow a minimum length to ensure a minimum level of security.</p> <p>Example: 512</p> <p>Permissible values: Any of the following:</p> <ul style="list-style-type: none"> <li>• 512</li> <li>• 1024</li> <li>• 2048</li> </ul> <p>Object type: Integer</p>
<code>maxSize</code>	<p>Specifies the maximum length for the key. Enter the length in bits. The default value is 2048.</p> <p>The current export restrictions imposed by the United States limit the exported versions of Netscape software to supporting public keys no longer than 512 bits. To allow keys certified by your server to work with software across national borders, you may want to enforce a maximum length of 512 bits.</p> <p>Example: 1024</p> <p>Permissible values: Any of the following:</p> <ul style="list-style-type: none"> <li>• 512</li> <li>• 1024</li> <li>• 2048</li> </ul> <p>Object type: Integer</p>



Table 15.8 Configurable parameters for the RSA key constraints policy and their values (Continued)

Parameter name	Description
<code>exponents</code>	<p>Limits the possible public exponent values. Use commas to separate different values.</p> <p>Some exponents are more widely used than others. The following exponent values are recommended for arithmetic and security reasons: 17 and 65537. Of these two values, 65537 is preferred. (This setting is mainly an issue if you are using your own software for generating key pairs. Key-generation programs in Netscape clients and servers use 3 or 65537.)</p> <p>Example: 17, 65537</p> <p>Permissible values: A combination of any of the following, separated by commas:</p> <ul style="list-style-type: none"> <li>• 3</li> <li>• 7</li> <li>• 17</li> <li>• 65537</li> </ul> <p>Object type: Integer</p>
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. To enable the rule, enter <code>true</code>; to disable the rule, enter <code>false</code>. By default, the rule is enabled.</p> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li>• <code>true</code></li> <li>• <code>false</code></li> </ul> <p>Object type: String</p>
<code>predicate</code>	<p>Specifies the predicate expression for this rule. If you want the rule to be applied to all certificate requests, leave the field blank. By default, the rule is applied to all certificate requests.</p> <p>Example: <code>certType==client</code></p> <p>Permissible values: See “Using Predicates in Policy Rules” on page 404.</p> <p>Object type: String</p>

## Validity Constraints Policy

The validity constraints policy applies to end-entity certificate enrollment requests. The policy enforces minimum and maximum validity periods for certificates and changes them if the policy is not met. Specifically, the policy imposes constraints on the following:

- The duration of a certificate's validity period (based on supported minimum and maximum validity periods).
- The lead and lag time for the beginning date and time (the `notBefore` and `notAfter` attributes in certificate requests) for the validity period; how far back into the front or back the `notBefore` date could go in minutes.

If this policy rule is enabled, the server applies the rule to the certificate request being processed, and then determines if the validity period in the request is acceptable. The rule checks two X.509 attributes of the certificate, the `notBefore` and `notAfter` time, which together indicate the total validity life of a certificate, to make sure that they conform to the configured ranges.

The rule checks that the value of the `notBefore` attribute in the request is not more than `leadTime` minutes in the future; the `leadTime` is a configurable parameter in the plug-in implementation. The ability to configure the value of the `leadTime` parameter in the policy rule allows you to prohibit end entities from requesting certificates whose validity starts too far in the future, and yet allows some amount of toleration of clock-skew problems. For example, if the current date and time is 01/15/1999 (mm/dd/YYYY) and 1:30 p.m., the value of the `notBefore` attribute is set to 3:00 p.m., and that the `leadTime` is 10 minutes, then the request would fail, because the validity requested begins more than 10 minutes in the future.

The rule also checks that the value of the `notBefore` attribute in the request is not more than `lagTime` minutes in the past. For example, if the current date and time is 01/15/1999 (mm/dd/yyyy) and 1:30 p.m., the value of the `notBefore` attribute is set to 1:15p.m., and the `lagTime` is set to 10 minutes, the request would fail because the user has requested a certificate 15 minutes in the past. Note that a request with `notBefore` set to 1:25 p.m. would have passed, however.

**Note** Currently, CRMF is the only enrollment format which allows an end entity to request a specific validity period.

It can be useful to restrict the length of the validity period for certificates issued by Certificate Management System. For example, if you want users to renew their certificates at least once a year, you can set the maximum validity period to one year. If you want to limit the frequency of certificate renewals to keep down administrative costs, you can set the minimum validity period to six months.

## Plug-in for Validity Constraints Policy

The plug-in module provided for the validity constraints policy is identified as follows:

```
com.netscape.certsrv.policy.ValidityConstraints
```

- In the configuration file, the implementation for this module is identified as follows; you can find it in the Policy section:

```
<subsystem>.Policy.impl.ValidityConstraints.class=com.netscape.certsrv.policy.ValidityConstraints
```

where <subsystem> indicates ca or ra (prefix identifying the subsystem)

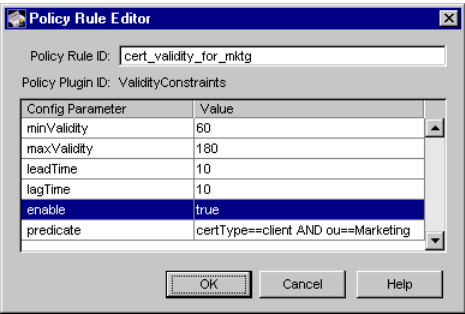
- In the CMS window, the implementation for this module is identified as follows; you can find it in the Policy Plugin Registration tab of the corresponding subsystem:

```
ValidityConstraints
```

## Configurable Parameters

Figure 15.6 shows how configurable parameters pertaining to the `ValidityConstraints` policy plug-in implementation are displayed in the CMS window.

Figure 15.6 Parameters and values for the ValidityConstraints module



The configuration shown in Figure 15.6 enforces a rule that all client certificates requested by end entities in an organizational unit (OU) called Marketing are valid for at least 60 days (two months) and require renewing after 180 days (six months).

Table 15.9 gives details about each of these parameters.

Table 15.9 Configurable parameters for the validity constraints policy and their values

Parameter name	Description
minValidity	<p>Specifies the minimum validity period, in days, for certificates. The default value is 180 days.</p> <p>Example: 60</p> <p>Permissible values: An integer greater than zero and less than the value specified by the maxValidity parameter</p> <p>Object type: Integer</p>
maxValidity	<p>Specifies the maximum validity period, in days, for certificates. The default value is 730 days.</p> <p>Example: 180</p> <p>Permissible values: An integer greater than zero and also greater than the value specified by the minValidity parameter</p> <p>Object type: Integer</p>

Table 15.9 Configurable parameters for the validity constraints policy and their values (Continued)

Parameter name	Description
leadTime	<p>Specifies the lead time, in minutes, for certificates. For a certificate renewal request to pass the renewal validity constraints policy, the value of the <code>notBefore</code> attribute in the certificate request must not be more than value of the <code>leadTime</code> parameter in the future, relative to the time when the policy rule is run. The default value is 10 minutes.</p> <p>The <code>notBefore</code> attribute value specifies the date on which the certificate validity begins; validity dates through the year 2049 are encoded as <code>UTCTime</code>, dates in 2050 or later are encoded as <code>GeneralizedTime</code>.</p> <p>Example: 10</p> <p>Permissible values: As applicable</p> <p>Object type: Integer</p>
lagTime	<p>Specifies the lag time, in minutes, for certificates. For a certificate renewal request to pass the renewal validity constraints policy, the value of the <code>notBefore</code> attribute in the certificate request must not be more than the value of the <code>lagTime</code> in the past, relative to the time when the policy is run. The default value is 10 minutes.</p> <p>The <code>notBefore</code> attribute value specifies the date on which the certificate validity ends; validity dates through the year 2049 are encoded as <code>UTCTime</code>, dates in 2050 or later are encoded as <code>GeneralizedTime</code>.</p> <p>Example: 10</p> <p>Permissible values: As applicable</p> <p>Object type: Integer</p>

Table 15.9 Configurable parameters for the validity constraints policy and their values (Continued)

Parameter name	Description
<code>beforeFix</code>	<p>Specifies the number of minutes to subtract from the current time when creating the value for the certificate's <code>notBefore</code> attribute. It can help some clients with incorrectly set clocks use the new certificate after downloading. For example, if the certificate is issued at 11:30 a.m. and the clock settings of the client into which the certificate is downloaded is 11:20 a.m., the certificate cannot be used for 10 minutes. Setting the value of the <code>beforeFix</code> parameter to 10 minutes would adjust the value of the <code>notBefore</code> parameter to 11:20 a.m.—thus making the certificate usable following the down load.</p> <p>Example: 5</p> <p>Permissible values: As applicable</p> <p>Object type: Integer</p>
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. To enable the rule, enter <code>true</code>; to disable the rule, enter <code>false</code>. By default, the rule is enabled.</p> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li>• <code>true</code></li> <li>• <code>false</code></li> </ul> <p>Object type: String</p>
<code>predicate</code>	<p>Specifies the predicate expression for this rule. If you want this rule to be applied to all certificate requests, leave the field blank. By default, the rule is applied to all certificate requests.</p> <p>Example: <code>certType==client AND ou==Marketing</code></p> <p>Permissible values: See “Using Predicates in Policy Rules” on page 404.</p> <p>Object type: String</p>

## Extension-Specific Policy Plug-in Modules

Since its initial publication, the X.509 standard for certificate formats has been amended to include additional information within a certificate. Version 3, the latest version, allows you to add additional information as certificate extensions. For example, Netscape applications (Netscape Navigator 3.0 or higher, and Enterprise Server 2.01 or higher) support an extension that specifies the type of certificate issued (such as client, server, or object signing).

Extension-specific policy plug-in modules help you configure Certificate Management System to set specific extensions on certificates it issues. When deciding whether to enable or disable any of the X.509 v3 certificate extensions, keep in mind that not all applications support X.509 v3 extensions. Among the applications that do support extensions, not all applications will recognize a given extension. For general guidelines on using extensions in certificates, see "Certificate Extensions" in Appendix B of *Netscape Certificate Management System Installation and Deployment Guide*.

Table 15.10 lists extension-specific policy plug-in modules supported out of the box. You can use these modules to define the policy rules for your Certificate Manager and Registration Manager.

Table 15.10 Extension-specific policy plug-in modules provided out of the box

Plug-in module name	Description
AuthorityKeyIdentifierExt	<p>Adds the <i>Authority Key Identifier</i> extension to a certificate depending on certificate type requested. For more information, see "Authority Key Identifier Extension Policy" on page 432.</p> <p>Only the Certificate Manager provides this plug-in module.</p>
BasicConstraintsExt	<p>Adds the <i>Basic Constraints</i> extension to a certificate depending on certificate type requested. For more information, see "Basic Constraints Extension Policy" on page 435.</p> <p>Only the Certificate Manager provides this plug-in module. You can add it to the Registration Manager's policy list.</p>

Table 15.10 Extension-specific policy plug-in modules provided out of the box (Continued)

Plug-in module name	Description
CRLDistributionPointsExt	<p>Adds the <i>CRL Distribution Point</i> extension to certificates. For more information, see “CRL Distribution Point Extension Policy” on page 438.</p> <p>Both the Certificate Manager and Registration Manager provide this plug-in module.</p>
KeyUsageExt	<p>Adds the <i>Key Usage</i> extension to a certificate depending on certificate type requested. For more information, see “Key Usage Extension Policy” on page 446.</p> <p>Both the Certificate Manager and Registration Manager provide this plug-in module.</p>
NSCertTypeExt	<p>Adds <i>Netscape Certificate Type</i> extensions to a certificate depending on certificate type requested. For more information, see “Netscape Certificate Type Extension Policy” on page 449.</p> <p>Both the Certificate Manager and Registration Manager provide this plug-in module.</p>
SubjectAltNameExt	<p>Adds the <i>Subject Alternate Name</i> extension depending on certificate type requested. For more information, see “Subject Alternate Name Extension Policy” on page 453.</p> <p>Both the Certificate Manager and Registration Manager provide this plug-in module.</p>
SubjectKeyIdentifierExt	<p>Adds the <i>Subject Key Identifier</i> extension depending on certificate type requested. For more information, see “Subject Key Identifier Extension Policy” on page 455.</p> <p>Both the Certificate Manager and Registration Manager provide this plug-in module.</p>

## Authority Key Identifier Extension Policy

The authority key identifier extension policy plug-in implementation is based on the Authority Key Identifier extension. This extension identifies the public key that corresponds to the private key used by a CA to sign certificates.



The key identifier set in the extension is the MD5 hash of the issuer's public key.

If enabled, this policy adds the authority key identifier extension to all certificates issued, unless the predicate specifies otherwise. You should consider using this extension for all CA certificates (root as well as subordinate) issued by Certificate Management System, especially when a Certificate Manager has multiple signing keys (either due to multiple concurrent key pairs or due to changeover).

For general guidelines on setting the authority key identifier extension, see "Certificate Extensions" in Appendix B of *Netscape Certificate Management System Installation and Deployment Guide*.

## Plug-in for Authority Key Identifier Extension Policy

The plug-in module provided for the authority key identifier extension policy is identified as follows:

```
com.netscape.certsrv.policy.AuthorityKeyIdExt
```

- In the configuration file, the implementation for this module is identified as follows; you can find it in the Policy section:

```
<subsystem>.Policy.impl.AuthorityKeyIdentifierExt.class=com.netscape.certsrv.policy.AuthorityKeyIdExt
```

where <subsystem> indicates ca or ra (prefix identifying the subsystem)

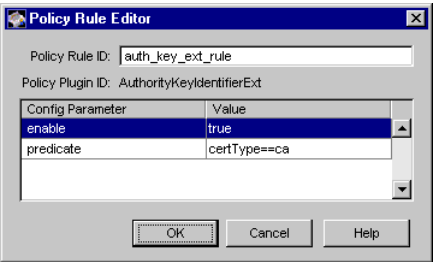
- In the CMS window, the implementation for this module is identified as follows; you can find it in the Policy Plugin Registration tab of the corresponding subsystem:

```
AuthorityKeyIdentifierExt
```

## Configurable Parameters

Figure 15.7 shows how the configurable parameters pertaining to the AuthKeyIDExt policy plug-in implementation are displayed in the CMS window.

Figure 15.7 Parameters and values for the AuthorityKeyIdentifierExt module



The configuration shown in Figure 15.7 enforces a rule that the authority key identifier extension must be set in all CA certificates.

Table 15.11 gives details about each of these parameters.

Table 15.11 Configurable parameters for the authority key identifier extension policy and their values

Parameter Name	Description
enable	<p>Specifies whether the rule is enabled or disabled. To enable the rule, enter <code>true</code>; to disable the rule, enter <code>false</code>. By default, the rule is enabled.</p> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li><code>true</code></li> <li><code>false</code></li> </ul> <p>Object type: String</p>
predicate	<p>Specifies the predicate expression for this rule. If you want this rule to be applied to all certificate requests, leave the field blank. By default, the rule is applied to all certificate requests.</p> <p>Example: <code>certType==ca</code></p> <p>Permissible values: See “Using Predicates in Policy Rules” on page 404.</p> <p>Object type: String</p>

## Basic Constraints Extension Policy

The basic constraints extension policy plug-in implementation is based on the Basic Constraints extension defined in X.509 and PKIX standard rfc 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>). This extension identifies whether the Certificate Manager is a CA. In addition, the extension is also used during the certificate chain verification process to identify CA certificates and to apply certificate chain-path length constraints.

**Note** The current PKIX standard requires that this extension be marked critical and that it appear in all CA certificates. The standard also recommends that the extension should not appear in end-entity certificates.

If enabled, this policy adds the basic constraints extension to all certificates being issued, unless specified otherwise by the predicate expression. To add the extension to subordinate CA certificates only, you must configure the predicate expression to be `certType==ca`.

For general guidelines on setting the basic constraints extension, see "Certificate Extensions" in Appendix B of *Netscape Certificate Management System Installation and Deployment Guide*.

### Plug-in for Basic Constraints Extension Policy

The plug-in module provided for the basic constraints extension policy is identified as follows:

```
com.netscape.certsrv.policy.BasicConstraintsExt
```

- In the configuration file, the implementation for this module is identified as follows; you can find it in the Policy section:

```
<subsystem>.Policy.impl.BasicConstraintsExt.class=com.  
netscape.certsrv.policy.BasicConstraintsExt
```

where <subsystem> indicates ca or ra (prefix identifying the subsystem)

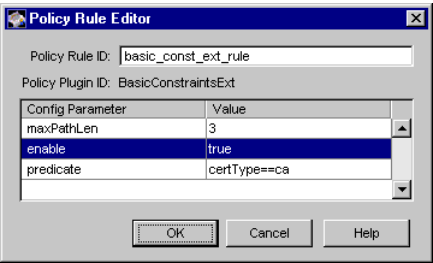
- In the CMS window, the implementation for this module is identified as follows; you can find it in the Policy Plugin Registration tab of the corresponding subsystem:

```
BasicConstraintsExt
```

### Configurable Parameters

Figure 15.8 shows how the configurable parameters pertaining to the `BasicConstraintsExt` policy plug-in implementation are displayed in the CMS window.

Figure 15.8 Parameters and values for the `BasicConstraintsExt` module



The configuration shown in Figure 15.8 enforces a rule that the basic constraints extension must be set in all CA certificates.

Table 15.12 gives details about each of these parameters.

Table 15.12 Configurable parameters for the basic constraints extension policy and their values

Parameter name	Description
MaxPathLen	<p>Specifies the path length. The value is the number of CA certificates that can be chained below (subordinate to) the subordinate CA certificate being issued. If this value is not set, it defaults to a value that is determined by the path length set on the issuer's certificate: it defaults to -1, if the issuer's path length is -1, or to one less than the issuer's path length, if the issuer's path length is greater than 0 (&gt;0).</p> <p>Note that the path length you specify effects the number of CA certificates to be used during certificate validation. The chain starts with the end-entity certificate being validated and moving up the chain. (The parameter has no effect if the extension is set in end-entity certificates.)</p> <p>Example: 3</p> <p>Permissible values: -1, 0, or n</p> <ul style="list-style-type: none"> <li>• -1 specifies unlimited or any number of subordinate CA certificates are allowed below the subordinate CA certificate being issued.</li> <li>• 0 specifies that no subordinate CA certificates are allowed below the subordinate CA certificate being issued.</li> <li>• n must be a specific number that is greater than zero. It specifies at the most n subordinate CA certificates are allowed below the subordinate CA certificate being used.</li> </ul> <p>Object type: Integer</p>
enable	<p>Specifies whether the rule is enabled or disabled. To enable the rule, enter <code>true</code>; to disable the rule, enter <code>false</code>. By default, the rule is enabled.</p> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li>• <code>true</code></li> <li>• <code>false</code></li> </ul> <p>Object type: String</p>

Table 15.12 Configurable parameters for the basic constraints extension policy and their values (Continued)

Parameter name	Description
predicate	<p>Specifies the predicate expression for this rule. If you want this rule to be applied to all certificate requests, leave the field blank. By default, the rule is applied to all certificate requests.</p> <p>Example: <code>certType==ca</code></p> <p>Permissible values: See “Using Predicates in Policy Rules” on page 404.</p> <p>Object type: String</p>

## CRL Distribution Point Extension Policy

The CRL distribution point extension policy plug-in implementation is based on the CRL Distribution Point extension. If enabled, this policy inserts a `CRLDistributionPoint` extension into each certificate being issued, unless the predicate expression specifies otherwise. Essentially, the extension identifies one or more CRL distribution points or locations where the CRL for the certificate (that includes the extension) can be obtained. For information on configuring a Certificate Manager to publish CRLs to different distribution points, see “CRL Issuing Points” on page 524.

Certificate Management System supports only two name forms for distribution points out of the box: X.500 Directory Name and URI. URIs described in this document support two CRL retrieval mechanisms: LDAP-based and HTTP-based.

Optionally, each distribution point may contain a set of reason flags, indicating what revocation reasons are covered by the CRL at that location. Also, the distribution point location can be relative to the location of the issuer. In this last case, the `issuerName` and `issuerType` parameters should be included to give the location of the issuer.

Note that you can configure the server to support any name form by changing the sample code provided for this purpose. You can find the code in a directory named `SDK` on the product CD. You can also download the samples from this site:

`http://home.netscape.com/eng/server/cms`

For general guidelines on setting the CRL distribution point extension in certificates, see "Certificate Extensions" in Appendix B of *Netscape Certificate Management System Installation and Deployment Guide*.

## Plug-in for CRL Distribution Point Extension Policy

The plug-in module provided for the CRL distribution point extension policy is identified as follows:

```
com.netscape.certsrv.policy.CRLDistributionPointsExt
```

- In the configuration file, the implementation for this module is identified as follows; you can find it in the Policy section:

```
<subsystem>.Policy.impl.CRLDistributionPointsExt.class=com.netscape.certsrv.policy.CRLDistributionPointsExt
```

where <subsystem> indicates ca or ra (prefix identifying the subsystem)

- In the CMS window, the implementation for this module is identified as follows; you can find it in the Policy Plugin Registration tab of the corresponding subsystem:

```
CRLDistributionPointsExt
```

## Configurable Parameters

Table 15.13 gives details about each of the configurable parameters pertaining to the CRLDistributionPointsExt policy plug-in implementation.

Table 15.13 Configurable parameters for the CRL distribution point extension policy and their values

Parameter name	Description
enable	<p>Specifies whether the rule is enabled or disabled. To enable the rule, enter <code>true</code>; to disable the rule, enter <code>false</code>. By default, the rule is enabled.</p> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"><li>• <code>true</code></li><li>• <code>false</code></li></ul> <p>Object type: String</p>
critical	<p>Specifies whether the extension should be marked critical. To mark the extension critical, enter <code>true</code>; to mark it noncritical, enter <code>false</code>. By default, it is marked noncritical; rfc 2459 specifies that the extension should not be marked critical (see <a href="http://www.ietf.org/rfc/rfc2459.txt">http://www.ietf.org/rfc/rfc2459.txt</a>).</p> <p>Example: <code>false</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"><li>• <code>true</code></li><li>• <code>false</code></li></ul> <p>Object type: String</p>



Table 15.13 Configurable parameters for the CRL distribution point extension policy and their values (Continued)

Parameter name	Description
<code>numPoints</code>	<p>Specifies the number of CRL distribution points to be contained or allowed in the extension. Note that each distribution point has its own configuration and you must specify the appropriate values for each of those parameters; otherwise the policy rule will return an error.</p> <p>Also note that the UI allows you to configure at the most two distribution points. However, there's no restriction on the total number of distribution points you can include in the extension, provided you update the CMS configuration file with the relevant information—that is, if you specify more than two distribution points, you must configure the remaining distribution points (of the total) by adding the corresponding parameters to the configuration file. For details, see “Adding Distribution Points to the Configuration File” on page 444.</p> <p>Example: 2</p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li>• 0 specifies that no distribution points can be contained in the extension.</li> <li>• n specifies the number of distribution points to be included in the extension; it must be a number greater than zero. For example, if you enter 3, three distribution points can be contained in the extension.</li> </ul> <p>Object type: Integer</p>

Table 15.13 Configurable parameters for the CRL distribution point extension policy and their values (Continued)

Parameter name	Description
pointName<x>	<p>Specifies the name of the CRL distribution point&lt;x&gt;, where x is an integer that is derived from the value of the numPoints parameter. For example, if you set the numPoints parameter to 3, x could be 0, 1, or 2.</p> <p>The name of the distribution point can be in any of the following formats:</p> <ul style="list-style-type: none"><li>• An X.500 Name in RFC1779 syntax, in which case the pointType attribute (shown below) must be DirectoryName.</li><li>• A URI, in which case the pointType attribute must be URI.</li><li>• An RDN which specifies a location relative to the CRL Issuer. In this case, the value of the pointType attribute must be RelativeToIssuer.</li></ul> <p>By default, the name can be an X.500 Directory Name and a URI. (You can configure the server to use any name form using the sample code provided.)</p> <p>Example:</p> <ul style="list-style-type: none"><li>• If the name is a URI, it would look similar to this: http://ca.somecorp.com/get/your/crls/here/</li><li>• If the name is an X.500 Directory Name, it would look similar to this: CN=CRLCentral,OU=Research Dept.,O=CertCorp,C=US</li></ul> <p>Permissible values: Any supported name forms</p> <p>Object type: String</p>
pointType<x>	<p>Specifies the type (for example, URI) of the CRL distribution point&lt;x&gt;, where x is an integer derived from the value of the numPoints parameter. For example, if you set the numPoints parameter to 2, x could be either 0 or 1.</p> <p>The value you specify for this parameter must correspond to that of the pointName parameter.</p> <p>Example: URI</p> <p>Permissible values: By default, DirectoryName and URI.</p> <p>Object type: String</p>

Table 15.13 Configurable parameters for the CRL distribution point extension policy and their values (Continued)

Parameter name	Description
<code>reasons&lt;x&gt;</code>	<p>Specifies the revocation reasons for the CRL maintained at distribution point&lt;x&gt;, where <code>x</code> is an integer derived from the value of the <code>numPoints</code> parameter. For example, if you set the <code>numPoints</code> parameter to 2, <code>x</code> could be either 0 or 1.</p> <p>Example: <code>keyCompromise</code></p> <p>Permissible values: A comma-separated list of the following constants:</p> <ul style="list-style-type: none"> <li>• <code>unused</code></li> <li>• <code>keyCompromise</code></li> <li>• <code>cACompromise</code></li> <li>• <code>affiliationChanged</code></li> <li>• <code>superseded</code></li> <li>• <code>cessationOfOperation</code></li> <li>• <code>certificateHold</code></li> </ul> <p>Object type: String</p>
<code>issuerName&lt;x&gt;</code>	<p>Specifies the name of the issuer that has signed the CRL maintained at distribution point&lt;x&gt;, where <code>x</code> is an integer derived from the value of the <code>numPoints</code> parameter. For example, if you set the <code>numPoints</code> parameter to 2, <code>x</code> could be either 0 or 1.</p> <p>The issuer name may be an X.500 Directory Name or a URI.</p> <ul style="list-style-type: none"> <li>• If the name is an X.500 Directory Name, the value of the <code>issuerType</code> parameter (below) must be <code>DirectoryName</code>.</li> <li>• If the name is a URI, the value of the <code>issuerType</code> parameter must be <code>URI</code>.</li> </ul> <p>Example: <code>OU=Research Dept.,O=CertCorp,C=US</code></p> <p>Permissible values: Any valid name</p> <p>Object type: String</p>

Table 15.13 Configurable parameters for the CRL distribution point extension policy and their values (Continued)

Parameter name	Description
<code>issuerType&lt;x&gt;</code>	<p>Specifies the type of the CRL issuer that has signed the CRL maintained at distribution point&lt;x&gt;, where <code>x</code> is an integer derived from the value of the <code>numPoints</code> parameter. For example, if you've set the <code>numPoints</code> parameter to 2, <code>x</code> could be either 0 or 1.</p> <p>Note that the value you specify for this parameter must map to the value of the corresponding <code>issuerName&lt;x&gt;</code> parameter.</p> <p>Example: <code>DirectoryName</code></p> <p>Permissible values: <code>DirectoryName</code> and <code>URI</code>.</p> <p>Object type: <code>String</code></p>
<code>predicate</code>	<p>Specifies the predicate expression for this rule. If you want this rule to be applied to all certificate requests, leave the field blank. By default, the rule is applied to all certificate requests.</p> <p>Example: <code>certType==Cisco-router</code></p> <p>Permissible values: See "Using Predicates in Policy Rules" on page 404.</p> <p>Object type: <code>String</code></p>

### Adding Distribution Points to the Configuration File

If you configure the CRL distribution point policy to contain more than two distribution points, you should add the configuration parameters pertaining to the remaining distribution points to the configuration file. The parameters you're required to add include the following:

- `<subsystem>.Policy.rule.CRLDistributionPoints.pointName<x>`
- `<subsystem>.Policy.rule.CRLDistributionPoints.pointType<x>`
- `<subsystem>.Policy.rule.CRLDistributionPoints.reasons<x>`
- `<subsystem>.Policy.rule.CRLDistributionPoints.issuerName<x>`
- `<subsystem>.Policy.rule.CRLDistributionPoints.issuerType<x>`

where, <subsystem> must be the prefix designated to the CMS subsystems—it is `ca` for the Certificate Manager and `ra` for the Registration Manager—to which the rule belongs.

To add CRL distribution point-specific parameters to the server's configuration:

1. Stop the CMS instance; see “Stopping Certificate Management System” on page 110.
2. Open the configuration file in a text editor; to locate the file, see “Locating the Configuration File” on page 71.
3. Add the configuration parameters to the file; see the configuration sample at the end of this procedure.
4. Save your changes, and close the configuration file.
5. Start the CMS instance; see “Starting Certificate Management System” on page 106.

The sample parameters shown below indicate how to create a CRL distribution point extension policy rule for a Certificate Manager. In the sample, note the following:

- The first distribution point (point 0) has no reason flags or issuer name.
- The second distribution point (point 1) has a directory (distinguished) name.
- The third distribution point (point 2) name is relative to the location of the CRL issuer, which is also given as a directory name.
- The second and third distribution points are differentiated by their reasons.

```
ca.Policy.rule.CRLDistributionPoints.numPoints=3
```

```
ca.Policy.rule.CRLDistributionPoints.pointName0=  
http://ca.somecorp.com/get/your/crls/here/
```

```
ca.Policy.rule.CRLDistributionPoints.pointType0=URI
```

```
ca.Policy.rule.CRLDistributionPoints.pointName1=  
CN=CRLCentral,OU=Research Dept.,O=CertCorp,C=US
```

```

ca.Policy.rule.CRLDistributionPoints.pointType1=DirectoryName
ca.Policy.rule.CRLDistributionPoints.reasons1=
    keyCompromise,cACompromise

ca.Policy.rule.CRLDistributionPoints.pointName2=CN=SubCN
ca.Policy.rule.CRLDistributionPoints.pointType2=
    RelativeToIssuer
ca.Policy.rule.CRLDistributionPoints.reasons2=
    superseded,cessationOfOperation
ca.Policy.rule.CRLDistributionPoints.issuerName2=
    OU=Research Dept.,O=CertCorp,C=US
ca.Policy.rule.CRLDistributionPoints.issuerType2=DirectoryName

```

## Key Usage Extension Policy

The key usage extension policy plug-in implementation is based on the Key Usage extension. This extension specifies for what purposes the key contained in the certificate should be used: whether the key should be used for data signing, key encipherment, or data encipherment. You can use this extension to restrict the usage of a key pair (or certificate). For example, you can restrict a certificate to be used for digital signature only.

If enabled, the policy adds the key usage extension to all certificates being issued, unless the predicate expression specifies otherwise. The bits set in the key usage extension are formed from the following HTTP input variables:

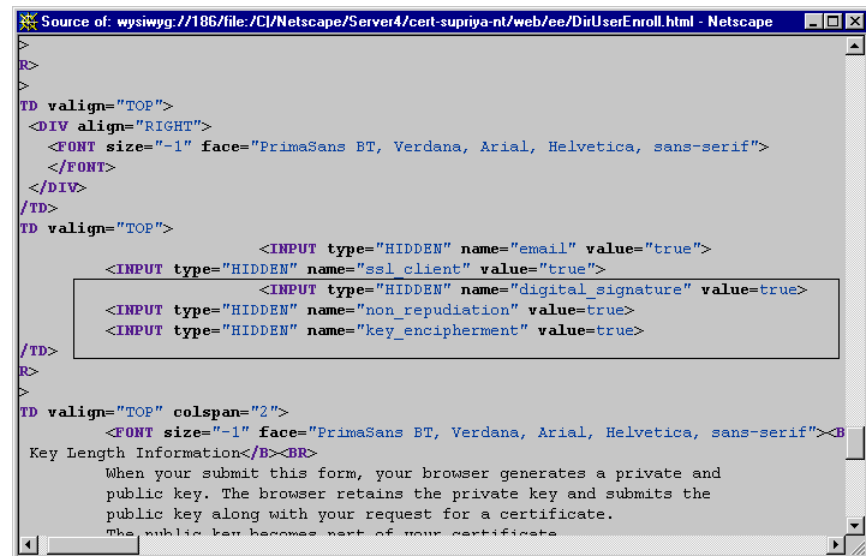
- `digital_signature` (bit 0)
- `non_repudiation` (bit 1)
- `key_encipherment` (bit 2)
- `data_encipherment` (bit 3)
- `key_agreement` (bit 4)
- `key_certsign` (bit 5)
- `crl_sign` (bit 6)
- `encipher_only` (bit 7)

- decipher\_only (bit 8)

The input variables are embedded as hidden values in the default enrollment forms—the directory-based enrollment form (DirUserEnroll.html), directory- and PIN-based enrollment form (DirPinUserEnroll.html), and manual enrollment form (ManUserEnroll.html). For details about these forms, see “Forms for Certificate Enrollment” on page 555.

Figure 15.9 shows the default directory-based enrollment form for end users with the information related to the key usage extension variables highlighted—it shows three of the total number of variables listed above, digital\_signature, non\_repudiation, and key\_encipherment, indicating that these bits be set in certificates requested using this form.

Figure 15.9 Key usage extension-specific variables in enrollment forms



Note that by default only a few variables are included in the form and all their values are set to *true*. You should make the appropriate modifications to suit your requirements. When adding new variables, the HTML input format must be as follows:

```
<input type="HIDDEN" name="variable_name" value=true>
```

where, *variable\_name* can be any of the HTTP input variables mentioned above.

For general guidelines on setting the key usage extension, see "Certificate Extensions" in Appendix B of *Netscape Certificate Management System Installation and Deployment Guide*.

**Plug-in for Key Usage Extension Policy**

The plug-in module provided for the key usage extension policy is identified as follows:

`com.netscape.certsrv.policy.KeyUsageExt`

- In the configuration file, the implementation for this module is identified as follows; you can find it in the Policy section:

```
<subsystem>.Policy.impl.KeyUsageExt.class=com.netscape.certsrv.policy.KeyUsageExt
```

where `<subsystem>` indicates `ca` or `ra` (prefix identifying the subsystem)

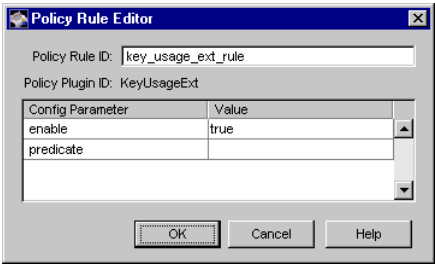
- In the CMS window, the implementation for this module is identified as follows; you can find it in the Policy Plugin Registration tab of the corresponding subsystem:

`KeyUsageExt`

**Configurable Parameters**

Figure 15.10 shows how the configurable parameters pertaining to the `KeyUsageExt` policy plug-in implementation are displayed in the CMS window.

Figure 15.10 Parameters and values for the `KeyUsageExt` module



The configuration shown in Figure 15.10 enforces a rule that the key usage extension must be set in all certificates.

Table 15.14 gives details about each of these parameters.



Table 15.14 Configurable parameters for the key usage extension policy and their values

Parameter name	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. To enable the rule, enter <code>true</code>; to disable the rule, enter <code>false</code>. By default, the rule is enabled.</p> <ul style="list-style-type: none"> <li>If you enable the rule, the server checks for the key usage extension bits specified by those HTTP input variables whose values are set to <i>true</i> (see Figure 15.9 on page 447), and sets them in the certificate being issued. If the values of all the input variables are set to <i>false</i> or if the variables are not embedded in the form, the server still sets the <code>digital_signature</code>, <code>non_repudiation</code>, and <code>key_encipherment</code> bits.</li> <li>If you disable the rule, the server ignores the key usage extension-specific input variables, irrespective of whether their values are set to true or false.</li> </ul> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li><code>true</code></li> <li><code>false</code></li> </ul> <p>Object type: String</p>
<code>predicate</code>	<p>Specifies the predicate expression for this rule. If you want this rule to be applied to all certificate requests, leave the field blank. By default, the rule is applied to all certificate requests.</p> <p>Example: <code>certType==client AND certType==server AND certType==ca</code></p> <p>Permissible values: See “Using Predicates in Policy Rules” on page 404.</p> <p>Object type: String</p>

## Netscape Certificate Type Extension Policy

The Netscape certificate type extension policy plug-in implementation is based on the Netscape Certificate Type extension. You can use this policy to limit the applications for a certificate. In other words, the extension identifies the type of

certificate; for example, it identifies whether the certificate is a CA certificate, server SSL certificate, client SSL certificate, object signing certificate, or S/MIME certificate.

- If the extension exists in a certificate, it limits the uses of the certificate to those specified.
- If the extension is not present, the certificate can be used for all applications except object signing.

This extension has no default value.

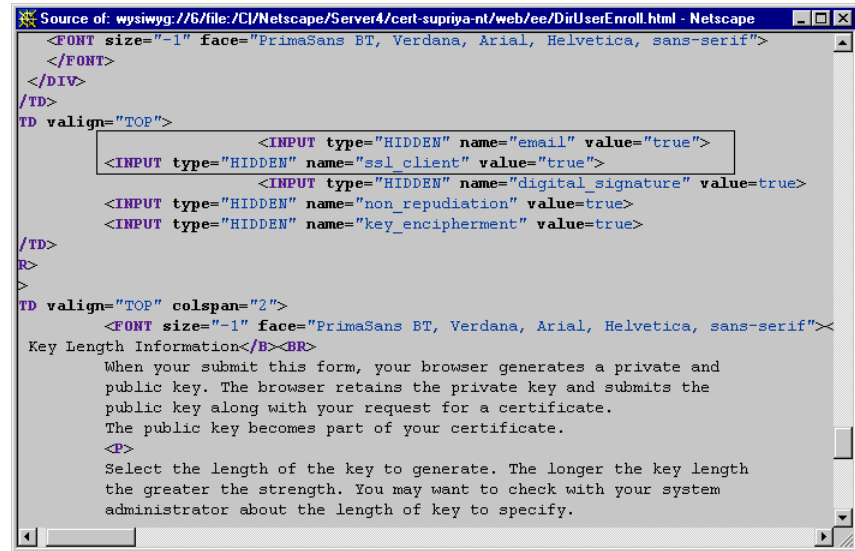
If enabled, the policy adds the Netscape certificate type extension to all certificates it issues, unless the predicate expression specifies otherwise. The bits set in the extension are formed from the following HTTP input variables:

- `ssl_client` (bit 0)
- `ssl_server` (bit 1)
- `email` (bit 2)
- `object_signing` (bit 3)
- Reserved for future use (bit 4)
- `ssl_ca` (bit 5)
- `email_ca` (bit 6)
- `object_signing_ca` (bit 7)

The input variables are embedded as hidden values in the default enrollment forms—the directory-based, directory- and PIN-based, and manual enrollment forms. For details about these forms, see “Forms for Certificate Enrollment” on page 555.

Figure 15.11 shows the default directory-based enrollment form for end users with the information related to the Netscape certificate type extension variables highlighted—it shows two of the total number of variables listed above, `ssl_client` and `email`, indicating that these bits be set in certificates requested using this form.

Figure 15.11 Netscape certificate type extension-specific variables in enrollment forms



Note that the default enrollment forms embed variables that are considered appropriate for the type of certificate, such as client, server, or CA, that can be requested using the form. For example, the server enrollment form embeds the `ssl_server` variable, whereas the subordinate CA (Certificate Manager) enrollment form embeds the `ssl_client`, `email_ca`, `ssl_ca` and `object_signing_ca` variables.

In general, the forms are set up so that you don't have to make any modifications. However, if there is a need to modify make sure that the HTML input format is as follows:

```
<input type="HIDDEN" value="true" name="variable_name">
```

where `variable_name` can be any of the variables listed above.

For general guidelines on setting Netscape certificate type extension, see "Certificate Extensions" in Appendix B of *Netscape Certificate Management System Installation and Deployment Guide*.

## Plug-in for Netscape Certificate Type Extension Policy

The plug-in module provided for the Netscape certificate type extension policy is identified as follows:

`com.netscape.certsrv.policy.NSCertTypeExt`

- In the configuration file, the implementation for this module is identified as follows; you can find it in the Policy section:

```
<subsystem>.Policy.impl.NSCertTypeExt.class=com.netscape.certsrv.policy.NSCertTypeExt
```

where `<subsystem>` indicates `ca` or `ra` (prefix identifying the subsystem)

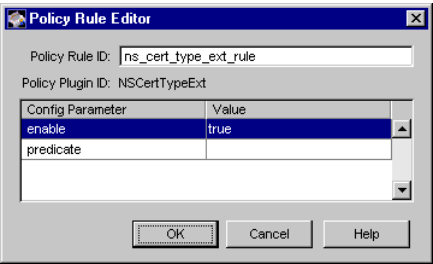
- In the CMS window, the implementation for this module is identified as follows; you can find it in the Policy Plugin Registration tab of the corresponding subsystem:

`NSCertTypeExt`

## Configurable Parameters

Figure 15.12 shows how the configurable parameters pertaining to the `NSCertTypeExt` policy plug-in implementation are displayed in the CMS window.

Figure 15.12 Parameters and values for the `NSCertTypeExt` module



The configuration shown in Figure 15.12 enforces a rule that the Netscape certificate type extension must be set in all certificates.

Table 15.15 gives details about each of these parameters.

Table 15.15 Configurable parameters for the Netscape certificate type extension policy and their values

Parameter name	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. To enable the rule, enter <code>true</code>; to disable the rule, enter <code>false</code>. By default, the rule is enabled.</p> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li>• <code>true</code></li> <li>• <code>false</code></li> </ul> <p>Object type: String</p>
<code>predicate</code>	<p>Specifies the predicate expression for this rule. If you want this rule to be applied to all certificate requests, leave the field blank. By default, the rule is applied to all certificate requests.</p> <p>Example: <code>certType==client AND certType==server AND certType==ca</code></p> <p>Permissible values: See "Using Predicates in Policy Rules" on page 404.</p> <p>Object type: String</p>

## Subject Alternate Name Extension Policy

The `SubjectAltNameExt` policy plug-in implementation is based on the Subject Alternate Name extension. If enabled, this policy checks the certificate request for a mail attribute, and if it is present, adds the subject alternate name extension to the certificate being issued. Both the built-in directory-based authentication modules can obtain a mail attribute from the authentication directory and set that attribute in the certificate request. For more information on the mail attribute, see the description for the `ldapAttributes` parameter in Table 9.1 on page 272 and Table 9.2 on page 277.

For general guidelines on setting the subject alternate name extension, see "Certificate Extensions" in Appendix B of *Netscape Certificate Management System Installation and Deployment Guide*.

### Plug-in for Subject Alternate Name Extension Policy

The plug-in provided for the subject alternate name extension policy is identified as follows:

`com.netscape.certsrv.policy.SubjectAltNameExt`

- In the configuration file, `CMS.cfg`, the implementation for this policy plug-in module is identified as follows; you can find it in the *Policy* section:

```
<subsystem>.Policy.impl.SubjectAltNameExt.class=com.netscape
.certsrv.policy.SubjectAltNameExt
```

where `<subsystem>` indicates `ca` or `ra` (prefix identifying the subsystem)

- In the CMS window, the implementation for this policy plug-in module is identified as follows; you can find it in the Policy Plugin Registration tab of the corresponding subsystem:

`SubjectAltNameExt`

### Configurable Parameters

Table 15.16 provides details for the configurable parameters pertaining to the `SubjectAltNameExt` policy plug-in implementation.

Table 15.16 Configurable parameters for the subject alternate name extension policy and their values

Parameter name	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. To enable the rule, enter <code>true</code>; to disable the rule, enter <code>false</code>. By default, the rule is enabled.</p> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"><li>• <code>true</code></li><li>• <code>false</code></li></ul> <p>Object type: String</p>

Table 15.16 Configurable parameters for the subject alternate name extension policy and their values (Continued)

Parameter name	Description
<code>predicate</code>	<p>Specifies the predicate expression for this rule. If you want this rule to be applied to all certificate requests, leave the field blank. By default, the rule is applied to all certificate requests.</p> <p>Example: <code>certType==client</code></p> <p>Permissible values: See “Using Predicates in Policy Rules” on page 404.</p> <p>Object type: String</p>

## Subject Key Identifier Extension Policy

The `SubjectKeyIDExt` policy plug-in implementation is based on the Subject Key Identifier extension. This extension provides a means of identifying certificates that contain a particular public key. In other words, this extension is used to uniquely identify a certificate from among several that have the same subject name.

The key identifier set in the extension is the MD5 hash of the subject’s public key information.

If enabled, this policy adds the subject key identifier extension to all certificates being issued, unless the predicate specifies otherwise. To facilitate chain building, this extension must appear in all conforming subordinate CA certificates (subordinate Certificate Managers’ CA signing certificates).

Therefore, the predicate expression is set to `certType==ca` by default—so that the extension is added to only subordinate CA certificates. However, you may modify the predicate expression to add this extension to other or all certificates.

If added to end-entity certificates, the subject key identifier extension provides a means for identifying certificates containing the particular public key used in an application. If an end entity has multiple certificates, especially from multiple CAs, the subject key identifier provides a means to quickly identify the set of certificates containing a particular public key. If you want to assist applications in identifying the appropriate end-entity certificate, you should modify the predicate expression to add this extension to all end-entity certificates.

For general guidelines on setting the subject key identifier extension, see "Certificate Extensions" in Appendix B of *Netscape Certificate Management System Installation and Deployment Guide*.

## Plug-in for Subject Key Identifier Extension Policy

The plug-in provided for the subject key identifier extension policy is identified as follows:

```
com.netscape.certsrv.policy.SubjectKeyIdExt
```

- In the configuration file, `CMS.cfg`, the implementation for this policy plug-in module is identified as follows; you can find it in the Policy section:

```
<subsystem>.Policy.impl.SubjectKeyIdIdentifierExt.class=com.netscape.certsrv.policy.SubjectKeyIdExt
```

where `<subsystem>` indicates `ca` or `ra` (prefix identifying the subsystem)

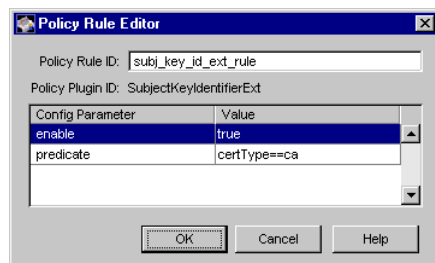
- In the CMS window, the implementation for this policy plug-in module is identified as follows; you can find it in the Policy Plugin Registration tab of the corresponding subsystem:

```
SubjectKeyIdIdentifierExt
```

## Configurable Parameters

Figure 15.13 shows how the configurable parameters pertaining to the `SubjectKeyIdExt` policy plug-in implementation are displayed in the CMS window.

Figure 15.13 `SubjectKeyIdExt` plug-in module: configurable parameters



The configuration shown in Figure 15.13 enforces a rule that the subject key identifier extension must be set in all CA certificates.



Table 15.17 provides details for each of these parameters.

**Table 15.17** Description of configuration parameters in the subject key identifier extension plug-in module

Parameter name	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. To enable the rule, enter <code>true</code>; to disable the rule, enter <code>false</code>. By default, the rule is enabled.</p> <p>Example: <code>true</code></p> <p>Permissible values: Either of the following:</p> <ul style="list-style-type: none"> <li>• <code>true</code></li> <li>• <code>false</code></li> </ul> <p>Object type: String</p>
<code>predicate</code>	<p>Specifies the predicate expression for this rule. If you want this rule to be applied to all certificate requests, leave the field blank. By default, the rule is applied to all certificate requests.</p> <p>Example: <code>certType==ca</code></p> <p>Permissible values: See “Using Predicates in Policy Rules” on page 404.</p> <p>Object type: String</p>



# Configuring Policies

Netscape Certificate Management System (CMS) provides a customizable policy framework for its main subsystems, the Certificate Manager, Registration Manager, and Data Recovery Manager. This chapter explains how to configure these subsystems to apply organizational and other policies on incoming certificate and key-related requests. The chapter also shows how policy plug-in implementations and rules (configured instances) appear in the configuration file.

Before reading this chapter, you should have read the chapter “Introduction to Policy” on page 401. In particular, you should be familiar with the various policy plug-in modules that come with Certificate Management System. If you are not, see “Built-in Policy Plug-in Modules” on page 409.

This chapter has the following sections:

- Policy Management (page 460)
- Managing Policy Rules (page 468)
- Managing Policy Plug-in Modules (page 479)

# Policy Management

You can manage both constraints and extensions policy rules in two ways:

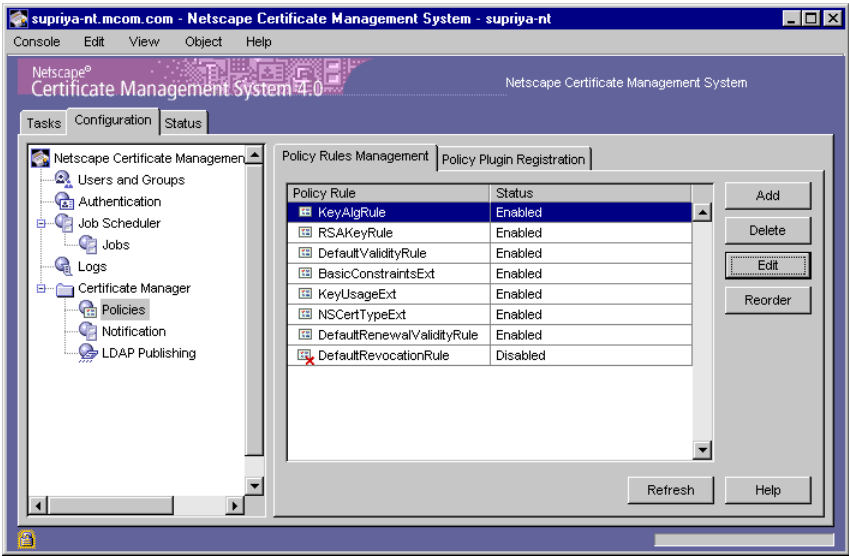
- By making changes to the authentication-specific parameters from the CMS window, explained in “Policy Management from the CMS Window” on page 460
- By editing the parameters in the configuration file, explained in “Policy Parameters in the Configuration File” on page 463

The recommended method is to use the CMS window.

## Policy Management from the CMS Window

The CMS window (as shown in Figure 16.1) provides the appropriate user interface to support policy management for each subsystem, the Certificate Manager, Registration Manager, and Data Recovery Manager.

Figure 16.1 Policy information in the CMS window



Under each subsystem tree node in the CMS window, you will find a **Policies** object. This object represents the policy processor specific to that subsystem. The processor represents the policy plug-in implementations and instances for the subsystem.

From this window you can accomplish the following operations:

- Configuration of currently registered policy plug-ins
- Registration of new (custom) policy plug-ins

The sections that follow describe the parts of the window from which you carry out these operations.

## Policy Rules Management Tab

The Policy Rules Management tab displays the current policy configuration for the selected subsystem. The tab lists the currently configured policy instances (or rules) for a subsystem, enabling you to manage them at a single place. From this tab you can add, modify, or delete rules, enable or disable individual rules, and change the order in which the rules get applied to an end-entity request.

**Add.** The add operation shows a list of registered policy plug-in modules from which you can select the one you want to configure. You configure an instance of the selected module with the help of the policy rule editor (see “Policy Rule Editor” on page 462). When you save the changes, the subsystem creates the rule and displays it in the list of policy rules. For instructions on adding new policy rules, see “Adding a Policy Rule” on page 468.

**Delete.** The delete operation allows you to remove unwanted policy rules from the CMS configuration. For instructions on deleting policy rules, see “Deleting a Policy Rule” on page 472.

**Edit/View.** The edit operation allows you to view and modify configuration parameter values associated with currently configured policy rules. You modify the parameter values with the help of the policy rule editor (see “Policy Rule Editor” on page 462). For instructions on modifying policy rules, see “Modifying a Policy Rule” on page 473.

**Reorder.** The reorder operation allows you to change the order of the policy rules a subsystem applies to an end-entity request. Enabled rules are applied in the order in which they appear; disabled rules are ignored. For instructions on reordering policy rules, see “Reordering Policy Rules” on page 477.

## Policy Rule Editor

The policy rule editor is designed to be generic. Its simple graphical interface enables you to create new policy rules and modify the configuration of an individual rule. When you are *adding* a new rule, the editor shows the configuration parameters pertaining to the plug-in module you selected. When you are *modifying* a rule, the editor shows the configuration parameters pertaining to the rule you selected.

All configurable parameters are displayed in the form of a table with two columns and multiple rows, each parameter occupying a row in the table. The left column lists the names of the configurable parameters; the right column is designated for entering the appropriate values. The ordering of the configurable parameters is irrelevant unless it is defined by the policy plug-in implementation.

You can also use the policy rule editor to change the status of a rule from enabled to disabled, or vice versa.

The policy rule editor provides normal save, cancel, and help functionality. You can specify *names* for policy rules, but only at the time of creating new ones; you cannot change the names later.

## Policy Plugin Registration Tab

The Policy Plugin Registration tab lists the currently registered policy plug-in implementations for the selected subsystem and gives you access to the window from which you can register new plug-in modules. On this tab you will find the names of registered plug-in modules listed on the left and the path to the Java class that implements the plug-in module listed on the right.

You can perform the following operations from this tab:

**Register.** This operation allows you to register a new policy plug-in module. You do this with the help of the policy registration editor (see “Policy Plug-in Registration Editor” on page 463).

When you save the changes, Certificate Management System loads the policy plug-in module and displays it in the list of currently registered plug-ins. For instructions on registering new authentication plug-in modules, see “Registering a Policy Plug-in Module” on page 479.

**Delete.** This operation allows you to remove unwanted policy plug-in modules from the CMS framework. For instructions on deleting policy plug-in modules, see “Deleting a Policy Plug-in Module” on page 481.

## Policy Plug-in Registration Editor

The policy plug-in registration editor allows you to register new plug-in implementations in a subsystem’s policy framework. Registering a new policy plug-in implementation involves specifying the name of the plug-in module and the full name of the Java class that implements the policy interface (implementation must be on the class path).

For example, you can add a policy implementation, named as follows, to the Data Recovery Manager’s policy framework:

```
com.netscape.policy.KeyArchivalPolicy
```

## Policy Parameters in the Configuration File

The sample shown in Figure 16.2 illustrates how policy-specific information appears in the configuration file. Keep the following points in mind:

- All policy-specific information, such as registered policy plug-in implementations, configured rules, and ordering, appear in the Policy section of the configuration file. If you have installed more than one subsystem in a CMS instance, for example Certificate Manager and Data Recovery Manager together, the configuration file will include policy sections that are specific to each of the subsystems that share the configuration.

You can identify policy pertaining to a subsystem by these suffixes:

- Certificate Manager by `ca`
- Registration Manager by `ra`

— Data Recovery Manager by kra

- Each registered policy plug-in module is identified by its implementation name.
- Each configured rule of a policy plug-in implementation is identified by the name you specified when creating it.
- You can create multiple rules out of an implementation; each rule must have a unique name. For details, see “Policy Plug-in Implementation and Rule” on page 466.



Figure 16.2 Policy-specific information for a Registration Manager in the configuration file

```

ra._000=##
ra._001=## RA Service
ra._002=##
ra.Policy.order=KeyAlgRule, RSAKeyRule, DefaultValidityRule, BasicConstraintsExt,
KeyUsageExt, NSCertTypeExt, DefaultRenewalValidityRule, DefaultRevocationRule

ra.Policy.impl._000=##
ra.Policy.impl._001=## Policy Implementations
ra.Policy.impl._002=##
ra.Policy.impl.BasicConstraintsExt.class=com.netscape.certsrv.policy.BasicConstraintsExt
ra.Policy.impl.DSAKeyConstraints.class=com.netscape.certsrv.policy.DSAKeyConstraints
ra.Policy.impl.DefaultRevocation.class=com.netscape.certsrv.policy.DefaultRevocation
ra.Policy.impl.KeyAlgorithmConstraints.class=com.netscape.certsrv.policy.KeyAlgorithmConstraints
ra.Policy.impl.KeyUsageExt.class=com.netscape.certsrv.policy.KeyUsageExt
ra.Policy.impl.NSCertTypeExt.class=com.netscape.certsrv.policy.NSCertTypeExt
ra.Policy.impl.RSAKeyConstraints.class=com.netscape.certsrv.policy.RSAKeyConstraints
ra.Policy.impl.RenewalValidityConstraints.class=com.netscape.certsrv.policy.RenewalValidityConstraints
ra.Policy.impl.ValidityConstraints.class=com.netscape.certsrv.policy.ValidityConstraints

ra.Policy.rule._000=##
ra.Policy.rule._001=## Policy Rules
ra.Policy.rule._002=##
ra.Policy.rule.BasicConstraintsExt.enable=true
ra.Policy.rule.BasicConstraintsExt.implName=BasicConstraintsExt
ra.Policy.rule.BasicConstraintsExt.predicate=

ra.Policy.rule.DSAKeyRule.enable=true
ra.Policy.rule.DSAKeyRule.implName=DSAKeyConstraints
ra.Policy.rule.DSAKeyRule.maxSize=2048
ra.Policy.rule.DSAKeyRule.minSize=512
ra.Policy.rule.DSAKeyRule.predicate=

ra.Policy.rule.DefaultRenewalValidityRule.enable=true
ra.Policy.rule.DefaultRenewalValidityRule.implName=RenewalValidityConstraints
ra.Policy.rule.DefaultRenewalValidityRule.maxValidity=365
ra.Policy.rule.DefaultRenewalValidityRule.minValidity=30
ra.Policy.rule.DefaultRenewalValidityRule.predicate=
ra.Policy.rule.DefaultRenewalValidityRule.renewalInterval=15

ra.Policy.rule.DefaultRevocationRule.enable=true
ra.Policy.rule.DefaultRevocationRule.implName=DefaultRevocation
ra.Policy.rule.DefaultRevocationRule.predicate=

ra.Policy.rule.DefaultValidityRule.enable=true
ra.Policy.rule.DefaultValidityRule.implName=ValidityConstraints
ra.Policy.rule.DefaultValidityRule.maxValidity=365
ra.Policy.rule.DefaultValidityRule.minValidity=30
ra.Policy.rule.DefaultValidityRule.predicate=

ra.Policy.rule.KeyAlgRule.algorithm=RSA
ra.Policy.rule.KeyAlgRule.enable=true
ra.Policy.rule.KeyAlgRule.implName=KeyAlgorithmConstraints
ra.Policy.rule.KeyAlgRule.predicate=

```

To change the configuration by editing the configuration file, follow the instructions in “Changing the Configuration by Editing the Configuration File” on page 72.

## Policy Plug-in Implementation and Rule

Policies are implemented as Java classes, which are then registered with the appropriate subsystem as plug-ins. You can use a given implementation of a policy plug-in module and configure multiple rules (instances) of it. Each rule must have a unique name (an alphanumeric string with no spaces) and can contain different input parameter values to apply to different requests. In other words, a given policy implementation can be shared by multiple configurations. You can also distinguish the applicability of configured instances by including appropriate names.

For example, with the help of predicates you can configure the `ValidityConstraints` plug-in module differently for users in two different organizational units (OUs). The figures that follow illustrate this. The same plug-in module, `ValidityConstraints`, has been used to configure two rules, *cert\_validity\_for\_sales* and *cert\_validity\_for\_mktg*, for employees in two organizational units (sales and marketing) by adding the predicates `ou==Sales` and `ou==Marketing`.

Figure 16.3 shows the two policy rules, both based on the same plug-in module, in the CMS window.

Figure 16.3 Separate certificate validity periods for users in two organizational units

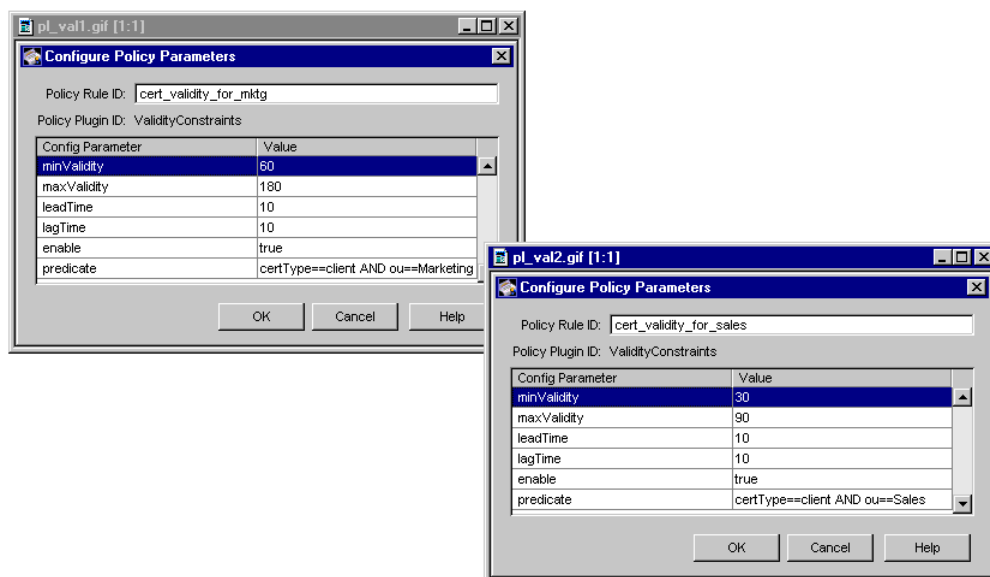


Figure 16.4 shows the two policy rules, both based on the same plug-in module, in the configuration file.

Figure 16.4 Separate certificate validity periods for users in two organizational units

```
## Policy rule for governing validity periods of certificates issued by a Registration Manager
## to users in an organizational unit (ou) calledSales

ra.Policy.rule.cert_validity_for_sales.implName=ValidityConstraints
ra.Policy.rule.cert_validity_for_sales.enable=true
ra.Policy.rule.cert_validity_for_sales.maxValidity=90
ra.Policy.rule.cert_validity_for_sales.minValidity=30
ra.Policy.rule.cert_validity_for_sales.leadTime=10
ra.Policy.rule.cert_validity_for_sales.lagTime=10
ra.Policy.rule.cert_validity_for_sales.predicate=certType==client AND ou==Sales

## Policy rule for governing validity periods of certificates issued by a Registration Manager
## to users in an organizational unit (ou) calledMarketing

ra.Policy.rule.cert_validity_for_mktg.implName=ValidityConstraints
ra.Policy.rule.cert_validity_for_mktg.enable=true
ra.Policy.rule.cert_validity_for_mktg.maxValidity=180
ra.Policy.rule.cert_validity_for_mktg.leadTime=10
ra.Policy.rule.cert_validity_for_mktg.lagTime=10
ra.Policy.rule.cert_validity_for_mktg.minValidity=30
ra.Policy.rule.cert_validity_for_mktg.predicate=certType==client AND ou==Marketing
```

# Managing Policy Rules

This section explains how to use the CMS window to perform the following operations:

- Adding a Policy Rule
- Deleting a Policy Rule
- Modifying a Policy Rule
- Reordering Policy Rules

For information on adding or changing policy-specific information in the configuration file, see “Policy Parameters in the Configuration File” on page 463.

## Adding a Policy Rule

Adding a policy rule to the CMS configuration involves creating a new instance of an already registered policy plug-in module, assigning a unique name (an alphanumeric string with no spaces) for the instance, and entering appropriate values for the parameters that define the plug-in implementation you want to create an instance of.

When you add a policy rule, the CMS configuration gets updated with policy-specific information. Keep the following points in mind:

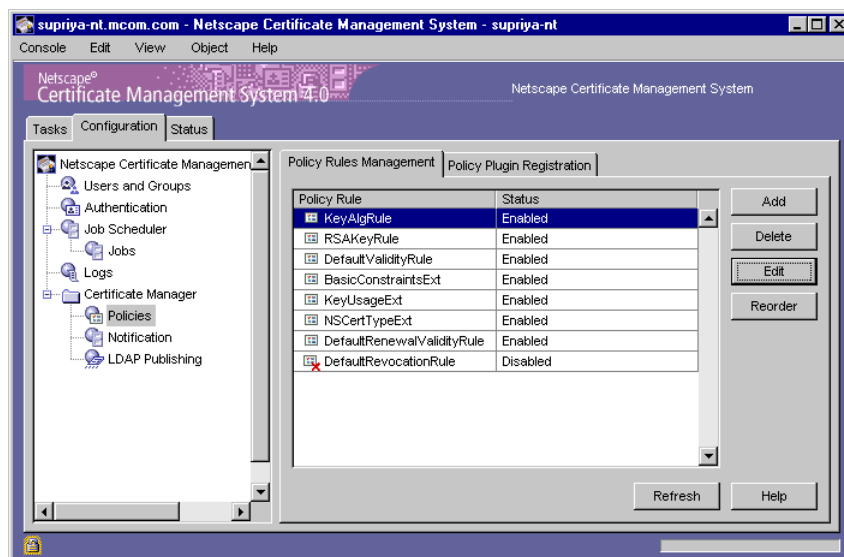
- The status of the rule, enabled or disabled, depends on the value (`true` or `false`) you enter for the `enable` parameter. A subsystem subjects end-entity requests only to rules that are enabled.
- The server does not automatically reorder rules. Be sure to change the order of the rule, if required. For information on reordering rules, see “Reordering Policy Rules” on page 477.

**Note** During installation, the Certificate Manager and Registration Manager automatically create policy rules that you would most likely want to use. For details, see “Modifying a Policy Rule” on page 473.

To add a new policy rule to the CMS configuration:

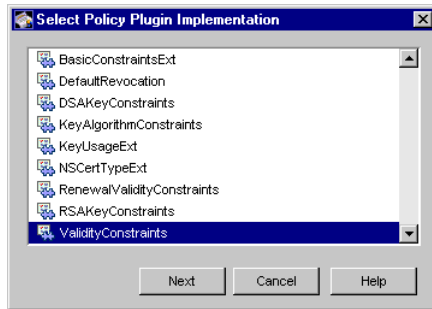
1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, choose the subsystem to which you want to add the new policy rule.
4. Click Policies.

The Policy Rules Management tab appears. It lists any currently configured policy rules. For information about this tab, see “Policy Rules Management Tab” on page 461.



5. Click Add.

The Select Policy Plugin Implementation window appears. It lists the currently registered policy plug-in modules.



6. Select a plug-in module.

The choices listed below are the ones provided out of the box with Certificate Management System. If you have registered any custom policy plug-ins, they too will be available for selection.

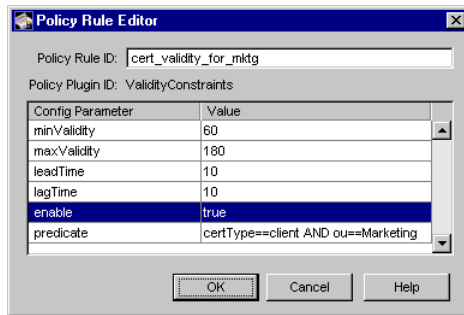
- **AuthorityKeyIdentifierExt**  
Policy for setting the Authority Key Identifier extension in certificates; see “Authority Key Identifier Extension Policy” on page 432.
- **BasicConstraintsExt**  
Policy for setting the Basic Constraints extension in certificates; see “Basic Constraints Extension Policy” on page 435.
- **CRLDistributionPointsExt**  
Policy for setting the CRL Distribution Point extension in certificates; see “CRL Distribution Point Extension Policy” on page 438.
- **DefaultRevocation**  
Policy for restricting certificate revocation; see “Default Revocation Policy” on page 411.
- **DSAKeyConstraints**  
Policy for restricting DSA key lengths; see “DSA Key Constraints Policy” on page 413.

- `KeyAlgorithmConstraints`  
Policy for restricting key algorithms; see “Key Algorithm Constraints Policy” on page 416.
- `KeyUsageExt`  
Policy for setting the Key Usage extension in certificates; see “Key Usage Extension Policy” on page 446.
- `NSCertTypeExt`  
Policy for setting the Netscape Certificate Type extension in certificates; see “Netscape Certificate Type Extension Policy” on page 449.
- `RenewalValidityConstraints`  
Policy for restricting certificate renewals; see “Renewal Validity Constraints Policy” on page 419.
- `RSAPKeyConstraints`  
Policy for restricting RSA key lengths; see “RSA Key Constraints Policy” on page 422.
- `SubjectAltNameExt`  
Policy for setting the Subject Alternate Name extension; see “Subject Alternate Name Extension Policy” on page 453.
- `SubjectKeyIdentifierExt`  
Policy for setting the Subject Key Identifier extension; see “Subject Key Identifier Extension Policy” on page 455.
- `ValidityConstraints`  
Policy for restricting certificate validity period; see “Validity Constraints Policy” on page 426.

For the purposes of this instruction, assume that you selected the `ValidityConstraints` module.

7. Click Next.

The Configure Policy Parameters window appears. It lists the configuration information required for this policy rule. For more information on how this window functions, see “Policy Rule Editor” on page 462.



8. In the Policy Rule ID field, type a unique name for this rule; be sure to use a name that will help you identify the rule.

For the name, be sure to use an alphanumeric string with no spaces.

9. In the configuration area, specify the required information by filling in parameter values in the text fields in the right column.

If you do not want to set any restrictions on a particular parameter, leave its value field blank.

10. Click OK.

You are returned to the Policy Rules Management tab.

11. If required, click Reorder and order the rules as appropriate following the information provided in “Reordering Policy Rules” on page 477.

## Deleting a Policy Rule

You can delete any unwanted policy rules from the CMS configuration. If you think you might need a rule in the future, instead of deleting it from the configuration you should disable it by setting the `enable` parameter value to `false`. In this way, you can avoid re-creating the rule in the future. Because all three subsystems subject end-entity requests only to rules that are currently



enabled (see “Policy Processor” on page 408), keeping unwanted rules in disabled state in the configuration does not affect policy decisions made by a subsystem.

To delete a policy rule from the CMS configuration:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, choose the subsystem to which the policy rule you want to delete belongs.
4. Click Policies.

The Policy Rules Management tab appears. It lists the currently configured policy rules. For information about this tab, see “Policy Rules Management Tab” on page 461.

5. In the Rule Name list, select the rule you want to delete and click Delete.
6. When prompted, confirm the delete action.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

## Modifying a Policy Rule

Modifying a policy rule involves changing its configuration parameter values; you cannot change the name of a rule. To change the name of a rule, create a new policy rule using the same policy plug-in module (that you used to create the rule you want to rename) with the same parameter values, and delete the old one.

As a part of modifying a rule, you can change its status from enabled to disabled or vice versa by changing the value of the `enable` parameter from `true` to `false` or vice versa. A subsystem subjects end-entity requests only to rules that are enabled.

During installation, the Certificate Manager and Registration Manager create default policy rules. Table 16.1 lists these rules. After installation, you must verify whether you want to use these rules, check how these rules are configured, and make the appropriate configuration changes. If you don't want to use a rule, delete it from the configuration following the instructions in “Deleting a Policy Rule” on page 472. If you want to create a new rule, follow the instructions in “Adding a Policy Rule” on page 468.

**Table 16.1** Default policy rules created for a Certificate Manager and Registration Manager

Policy rule name	Policy plug-in module name
AuthorityKeyIdentifierExt	AuthorityKeyIdentifierExt See “Authority Key Identifier Extension Policy” on page 432.
BasicConstraintsExt	BasicConstraintExt See “Basic Constraints Extension Policy” on page 435.
DefaultRenewalValidityRule	RenewalValidityConstraints See “Renewal Validity Constraints Policy” on page 419.
DefaultRevocationRule	DefaultRevocation See “Default Revocation Policy” on page 411.
DefaultValidityRule	ValidityConstraints See “Validity Constraints Policy” on page 426.
DSAKeyRule	DSAKeyConstraints See “DSA Key Constraints Policy” on page 413.
KeyAlgRule	KeyAlgorithmConstraints See “Key Algorithm Constraints Policy” on page 416.

Table 16.1 Default policy rules created for a Certificate Manager and Registration Manager

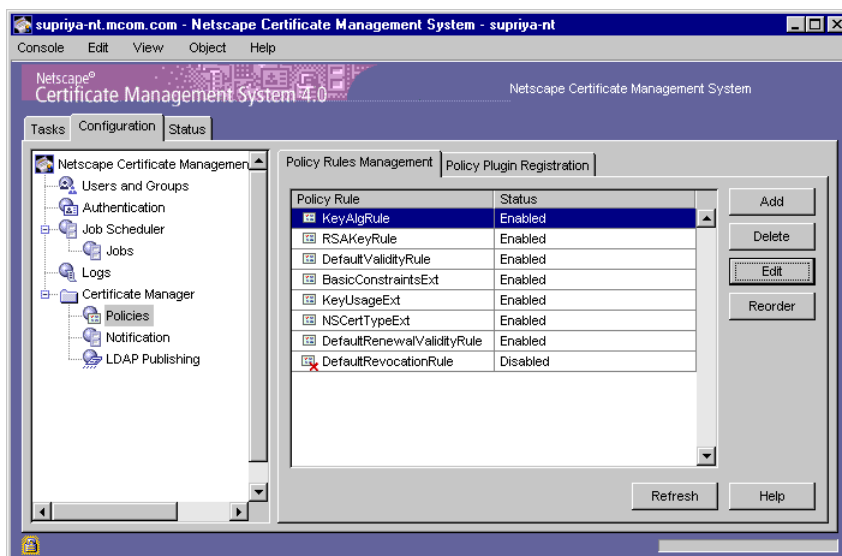
Policy rule name	Policy plug-in module name
KeyUsageExt	KeyUsageExt See “Key Usage Extension Policy” on page 446.
NSCertTypeExt	NSCertTypeExt See “Netscape Certificate Type Extension Policy” on page 449.
RSAPKeyRule	RSAPKeyConstraints See “RSA Key Constraints Policy” on page 422.
SubjectAltNameExt	SubjectAltNameExt See “Subject Alternate Name Extension Policy” on page 453.
SubjectKeyIdentifierExt	SubjectKeyIdentifierExt See “Subject Key Identifier Extension Policy” on page 455.

To modify a policy rule in the CMS configuration:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, choose the subsystem to which the policy rule you want to modify belongs.

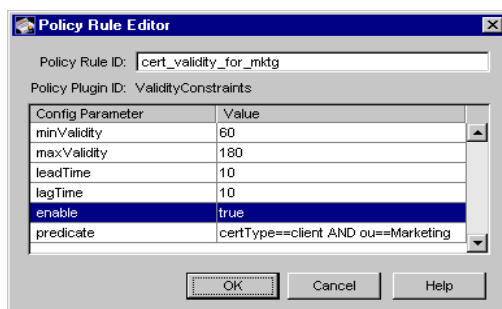
4. Click Policies.

The Policy Rules Management tab appears. It lists the currently configured policy rules. For information about this tab, see “Policy Rules Management Tab” on page 461.



5. In the Rule Name list, select the rule you want to modify and click Edit.

The Configure Policy Parameters window appears, showing how this rule is currently configured. An example is shown below. For more information on how this window functions, see “Policy Rule Editor” on page 462.



6. Make the necessary changes by filling in parameter values in the text fields in the right column.

If you do not want to set any restrictions on a particular parameter, leave its value field blank.

7. Click OK.

You are returned to the Policy Rules Management tab.

8. If required, click Reorder and order the rules as appropriate following the information provided in “Reordering Policy Rules” on page 477.

## Reordering Policy Rules

For maintaining priority levels, Certificate Management System supports a linear list of policy rules in increasing order of priority. This means that for a given policy category in the configuration file, a policy configuration with a lower priority precedes one with a higher priority. This simple linear listing avoids the need to have explicit locking on request attributes to prevent conflicting changes. By ordering the rules, you introduce a concurrency control whereby a higher-priority rule configuration overwrites any changes made by a lower-priority rule configuration that precedes it.

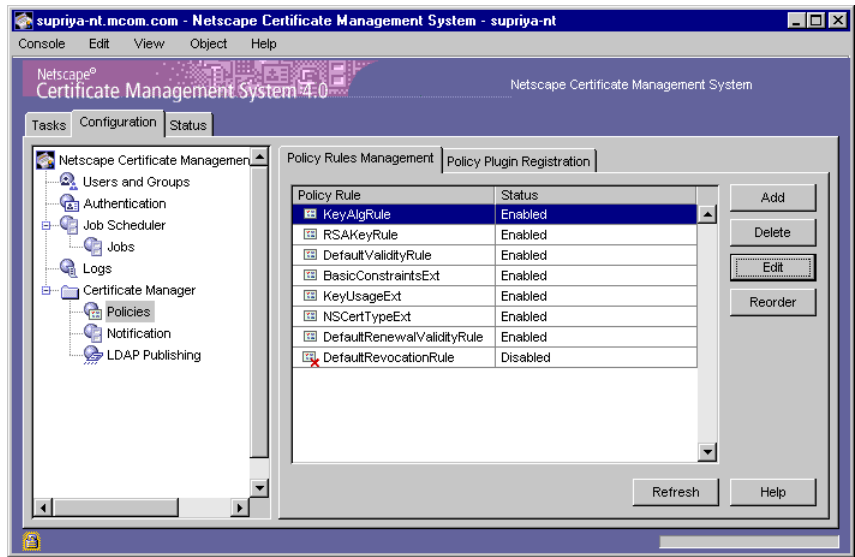
You may want to specify policies at different priority levels for the same operation depending on the end-entity information. For example, authentication policies, if any, need to precede others in the list.

To reorder policy rules in the CMS configuration:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, choose the subsystem.

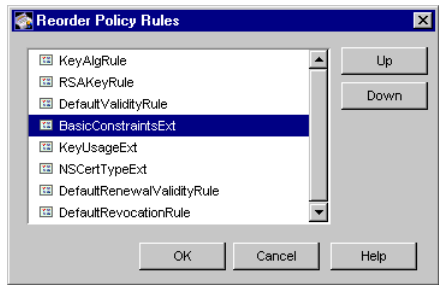
4. Click Policies.

The Policy Rules Management tab appears. It lists the currently configured policy rules. For more information about this tab, see “Policy Rules Management Tab” on page 461.



5. Click Reorder.

The Reorder Policy Rules window appears. It lists the currently registered policy plug-in modules in the order in which they are executed by the subsystem; the server executes the rules on a first-come-first-served basis, overwriting the configuration determined by the previous rule, if any.



6. To change the order of a rule, select it in the list and click the Up or Down button, as appropriate.

7. When you have the correct order, click OK.

You are returned to the Policy Rules Management tab.

8. To view the updated configuration, click Refresh.

## Managing Policy Plug-in Modules

This section explains how to use the CMS window to perform the following operations:

- Registering a Policy Plug-in Module
- Deleting a Policy Plug-in Module

For information on adding or changing policy-specific information in the configuration file, see “Policy Parameters in the Configuration File” on page 463.

### Registering a Policy Plug-in Module

You can register custom policy plug-in modules from the CMS window. Before registering a plug-in module, be sure to put the Java class for the plug-in in the `classes` directory.

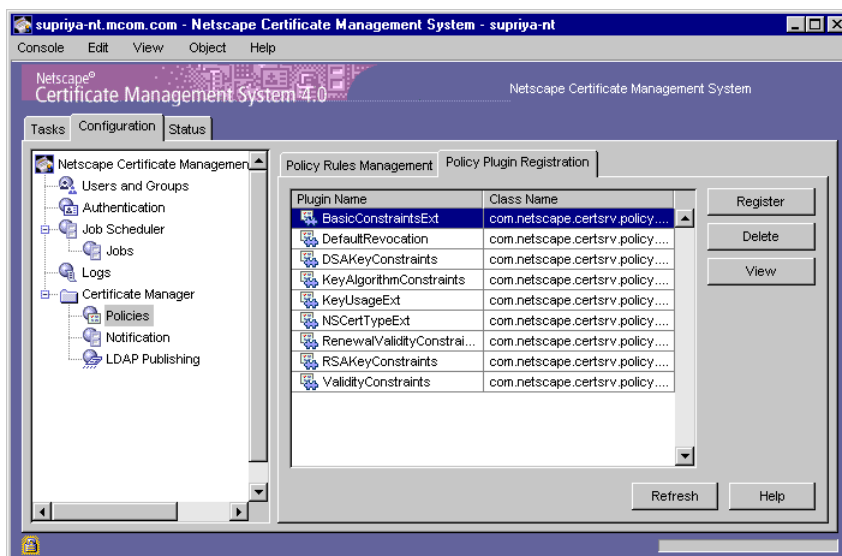
To register a policy plug-in module in a subsystem’s policy framework:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, choose the subsystem that will use the plug-in module you want to register.
4. Click Policies.

The Policy Rules Management tab appears. It lists the currently configured policy rules.

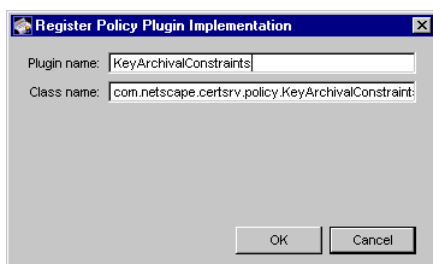
- Click the Policy Plugin Registration tab.

The Policy Plugin Registration tab appears. It lists the currently registered policy plug-in modules. For information about this tab, see “Policy Plugin Registration Tab” on page 462.



- Click Register.

The Register Policy Plugin Implementation window appears. For information on how this window works, see “Policy Plug-in Registration Editor” on page 463.



- Specify information as appropriate:

**Plugin name.** Type the name of the plug-in.



**Class name.** Type the full name of the class for this plug-in—that is, the path to the implementing Java class. If this class is part of a package, be sure to include the package name. For example, if you are registering a class named `myPolicy` and if this class is in a package named `com.myCompany`, type `com.myCompany.myPolicy`.

8. Click OK.

You are returned to the Policy Plugin Registration tab.

9. To view the updated configuration, click Refresh.

## Deleting a Policy Plug-in Module

You can delete unwanted policy plug-in modules using the CMS window. Before deleting a plug-in module, be sure to delete all the policy rules that are based on this plug-in module; see “Deleting a Policy Rule” on page 472.

To delete a policy plug-in module from a subsystem’s policy framework:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, choose the subsystem that registers the plug-in module you want to delete.
4. Click Policies.

The Policy Rules Management tab appears. It lists the currently configured policy rules.

5. Click the Policy Plugin Registration tab.

The Policy Plugin Registration tab appears. It lists the currently registered policy plug-in modules. For information about this tab, see “Policy Plugin Registration Tab” on page 462.

6. In the Plugin Name list, select the plug-in you want to delete and click Delete.
7. When prompted, confirm the delete action.



# *LDAP Publishing*

*Chapter 17* Introduction to LDAP Publishing

*Chapter 18* Configuring Subsystems for LDAP Publishing

*Chapter 19* Publishing CRLs



# Introduction to LDAP Publishing

Large corporations typically use Lightweight Directory Access Protocol (LDAP) directories, such as Netscape Directory Server, to store and manage corporatewide data, including user and group information and network resource data. If you have deployed an LDAP-compliant directory, you can configure Netscape Certificate Management System (CMS) to automatically publish your end-entity certificate-related information to that directory, called a *publishing directory*.

If you have configured Certificate Management System to employ directory-based authentication, consider publishing end-entity certificates to the same directory. The advantage of publishing certificates and certificate revocation lists (CRLs) to the directory used for authentication is that you can keep your user's certificate-related information with the rest of the user information (see Figure 17.1).

This chapter explains how Certificate Management System works with the publishing directory and outlines the kind of directory configuration required for publishing certificate-related information.

The chapter has the following sections:

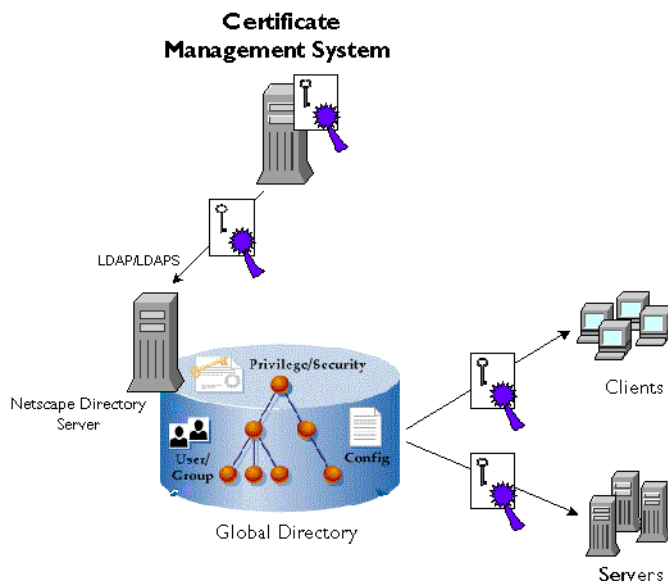
- What Is LDAP Publishing? (page 486)
- Timing of Directory Updates (page 488)
- Directory Update Process (page 490)

- Directory Schema Requirements (page 499)
- Directory Synchronization (page 501)

## What Is LDAP Publishing?

In Certificate Management System, LDAP publishing refers to the ability of a Certificate Manager or Registration Manager to publish certificates, CRLs, and other certificate-related objects to a directory using the LDAP protocol. Configuring the Certificate Manager or Registration Manager for LDAP publishing is optional—you can turn this feature off without affecting any of the certificate-management operations handled by the Certificate Manager or Registration Manager.

Figure 17.1 Publishing certificates and CRLs to a directory for distribution



You can configure Certificate Management System to automatically publish certificates to the directory every time a certificate is issued or at a predetermined interval—for example, every day. Privileged users (administrators and agents) can also manually initiate the LDAP publishing process.

Figure 17.2 illustrates LDAP publishing by the Certificate Manager when a certificate requested via the manual enrollment process is issued.

Figure 17.2 Publishing by a Certificate Manager

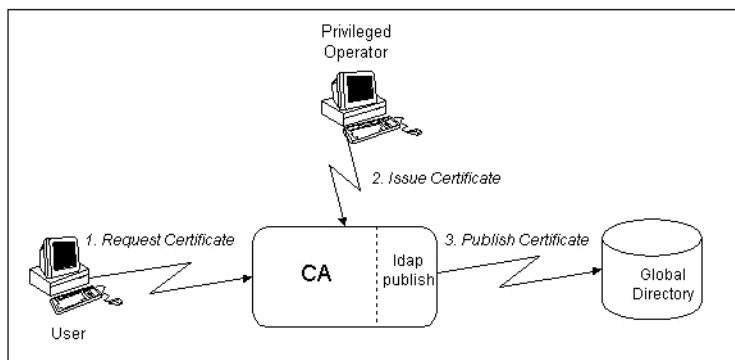
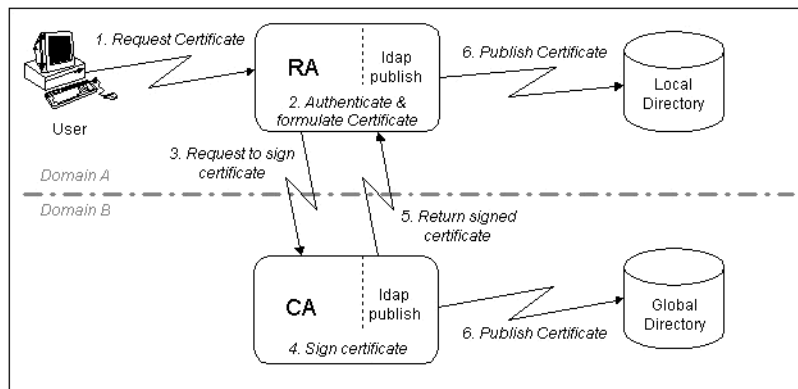


Figure 17.3 illustrates a configuration in which both the Registration Manager and Certificate Manager publish to separate directories (both are configured for LDAP publishing). The publishing process is initiated by a certificate issuance operation from a Registration Manager.

Figure 17.3 Publishing by a Registration Manager and Certificate Manager



# Timing of Directory Updates

If Directory Server is properly configured to work with Certificate Management System, any changes to certificate information in Certificate Management System are automatically updated in the directory.

Updates take place at specific times:

- When Certificate Management System starts up, it publishes the Certificate Manager's certificate to the directory.
- When Certificate Management System issues a new certificate, it publishes that certificate to the directory.
- When Certificate Management System revokes a certificate, it removes that certificate from the directory and updates the CRL.
- When the CRL is created or updated, the list is published to the directory. See "Publishing CRLs" on page 523.

The Certificate Manager and Registration Manager publish specific objects to the directory. For details, see "Objects Published by the Certificate Manager" on page 489 and "Objects Published by the Registration Manager" on page 490.

Certificate Management System fails to publish to the directory in the following cases:

- If a user entry is not present or if an entry cannot be found to publish the certificate.
- If the directory schema doesn't include the appropriate attributes. To configure the directory for LDAP publishing, see "Directory Schema Requirements" on page 499. Note that Certificate Management System publishes to the `userCertificate;binary` attribute, which is an LDAP v3 standard. Unless you are using a non-standards compliant directory, this situation shouldn't arise.
- When the directory is unreachable due to maintenance or network/system failures.



# Objects Published by the Certificate Manager

By default, the Certificate Manager publishes specific objects to the directory configured for LDAP publishing. These objects and associated details are listed in Table 17.1. To configure the Certificate Manager for LDAP publishing, see “Configuring Subsystems for LDAP Publishing” on page 503.

The Certificate Manager’s LDAP publishing action happens as a separate transaction from any certificate operation (such as issuance); the certificate operation is not affected by whether the object was successfully published or not.

Table 17.1 Details of objects published by the Certificate Manager

Object	Action	Timing	LDAP entry	LDAP attribute	Object format
End-entity certificate	Publish	Occurs when a certificate is issued or renewed	End entity's entry	<code>userCertificate;binary</code>	DER encoded binary blob
End-entity certificate	Unpublish (remove)	Occurs when a certificate is revoked or expired	End entity's entry	<code>userCertificate;binary</code>	DER encoded binary blob
CA certificate	Publish	Occurs when the Certificate Manager is started	CA's entry	<code>caCertificate;binary</code>	DER encoded binary blob
CRL (full)	Publish (replace)	Occurs when a new CRL is generated	CA's entry	<code>certificateRevocationList;binary</code>	DER encoded binary blob

# Objects Published by the Registration Manager

By default, the Registration Manager publishes specific objects to the directory configured for LDAP publishing. These objects and the associated details are listed in Table 17.2. To configure the Certificate Manager for LDAP publishing, see “Configuring Subsystems for LDAP Publishing” on page 503.

As in LDAP publishing by the Certificate Manager, the result of any certificate operation is not affected by the Registration Manager's LDAP publishing action.

Table 17.2 Details of objects published by the Registration Manager

Object	Action	Timing	LDAP entry	LDAP attribute	Object format
End-entity certificate	Publish	Occurs when the Certificate Manager returns a signed certificate for the Registration Manager's issuance or renewal request	End entity's entry	userCertificate;binary	DER encoded binary blob
End-entity certificate	Unpublish (remove)	Occurs when the Certificate Manager responds success to a revocation request	End entity's entry	userCertificate;binary	DER encoded binary blob

## Directory Update Process

When Certificate Management System is requested to issue a certificate or to update certificate information, it automatically publishes or updates the certificate information for the corresponding entry in the configured LDAP directory. To locate the correct directory entry and publish certificate

information to it, Certificate Management System uses object mapping and publishing rules. For details, see “Object-Mapping Rules” on page 491 and “Object-Publishing Rules” on page 498.

Similarly, when you revoke a certificate, Certificate Management System automatically deletes the corresponding certificate from the directory.

## Object-Mapping Rules

Before Certificate Management System can publish or update a certificate in the directory, it must first find the directory entry that needs to be updated. In order to find the correct directory entry to update, Certificate Management System needs to present the directory (Directory Server) with appropriate search criteria so that it can initiate an LDAP search operation; the server considers the search successful only if Directory Server returns a single LDAP entry that exactly matches the search criteria.

A Certificate Manager or Registration Manager uses object-mapping rules to find the directory entry that needs to be updated. The mapping rules help the server to build appropriate LDAP search criteria (that results in locating the exact entry that needs to be updated) and present it to the LDAP directory. Object-mapping rules are implemented as Java classes and registered in the CMS configuration.

## Built-in Mapper Classes

*Mapper* classes map an object to an entry in the directory. By default, Certificate Management System provides a single mapper class for mapping certificates to entries in the publishing directory. The rule implemented by this class enables a Certificate Manager or Registration Manager to map an X.509 certificate to an LDAP entry in these ways:

- By formulating the entry's distinguished name (DN) from components (such as CN, OU, O, and C) in the certificate's subject name and using it as the search DN to locate the entry in the directory. For details, see “How Mapping by DN Components Works” on page 496.
- By using a search filter for searching the certificate's subject name in an entry.

In general, the rule takes DN components to build the search DN. The rule also takes an optional root search DN. The server uses the DN components to form an LDAP entry to begin a subtree search and the filter components to form a search filter for the subtree. If none of the DN components are configured, the server uses the base DN for the subtree. If the base DN is null and none of the DN components match, an error is returned. If none of the DN components and filter components match, an error is returned. If the filter components are null, a base search is performed.

The Java class provided for mapping certificates to directory entries based on DN components is identified as follows:

```
com.netscape.certsrv.ldap.LdapCertCompsMap
```

- In the CMS window, the implementation for this class is identified as follows; you can find it in the LDAP configuration section of the user interface:

```
LdapCertCompsMap
```

- In the configuration file, the implementation for this class is identified as follows; you can find it in the LDAP section:

```
<subsystem>.ldappublish.mapConfig.mapper.class=com.netscape.certsrv.ldap.LdapCertCompsMap
```

where <subsystem> indicates ca or ra (a prefix identifying the subsystem)

You can also write your own mapper classes by implementing the following Java interface:

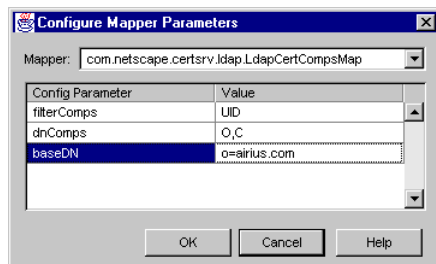
```
com.netscape.certsrv.ldappublish.ILdapMapper
```

For more information about this interface, see the SDKs directory on the product CD.

## Configurable Parameters in the LdapCertCompsMap Class

Figure 17.4 shows how the configurable parameters pertaining to the LdapCertCompsMaps mapper class are displayed in the CMS window.

Figure 17.4 Configurable parameters for the LdapCertMapComps mapper class



With this configuration, a Certificate Manager or Registration Manager maps its certificates with the ones in the LDAP directory by using the `dnComps` values to form a DN and the `filterComps` values to form a search filter for the subtree.

- If the formed DN is null, the server uses the `baseDN` value for the subtree. If both the formed DN and base DN are null, the server logs an error.
- If the filter is null, the server uses the `baseDN` value for the search. If both the filter and base DN are null, the server logs an error.

Table 17.3 provides details for each of these parameters.

Table 17.3 Configuration parameters for mapping and publishing certificates and CRLs to the directory

Parameter name	Description
filterComps	<p>The server uses the <code>filterComps</code> values to form an LDAP search filter for the subtree. The server constructs the filter by gathering values for these attributes from the user's DN in the certificate; it uses the filter to search for and match entries in the LDAP directory.</p> <p>If the server finds one or more entries in the LDAP directory that match the user's information gathered from the certificate, the search is successful and the server optionally performs a verification. For example, if <code>filterComps</code> is set to use the email and user ID attributes (<code>filterComps=e, uid</code>), the server searches the directory for an entry whose values for email and user ID match the end user's information gathered from the client certificate.</p> <p>Email addresses and user IDs are good filters because they are usually unique entries in the directory. Keep in mind that email is not always included in the certificate subject name. The filter needs to be specific enough to match one and only one entry in the LDAP database.</p> <p>Example: <code>UID</code></p> <p>Permissible values: Valid directory attributes (in the certificate DN) separated by commas.</p> <p>The attribute names for the filters need to be attribute names from the certificate, not from ones in the LDAP directory. For example, most certificates have an <code>E</code> attribute for the user's email address; LDAP calls that attribute <code>mail</code>.</p> <p>Object type: String</p>

Table 17.3 Configuration parameters for mapping and publishing certificates and CRLs to the directory (Continued)

Parameter name	Description
<code>dnComps</code>	<p>Use this parameter to specify where in the LDAP directory Certificate Management System should start searching for entries that match the end entity's information (that is, the owner of the certificate). The server uses the <code>dnComps</code> values to form an LDAP entry to begin a subtree search.</p> <p>The server gathers values for these attributes from the certificate subject name and uses the values to form an LDAP DN, which then determines where in the LDAP directory the server starts its search. For example, if you set <code>dnComps</code> to use the <code>o</code> and <code>c</code> attributes of the DN, the server starts the search from the <code>o=&lt;org&gt;</code>, <code>c=&lt;country&gt;</code> entry in the LDAP directory, where <code>&lt;org&gt;</code> and <code>&lt;country&gt;</code> are replaced with values from the DN in the certificate.</p> <p>If the <code>dnComps</code> entry is present but has no value, the server searches the entire LDAP tree for entries matching the filter specified by <code>filterComps</code> parameter values.</p> <p>Example: <code>O,C</code></p> <p>Permissible values: Valid DN components or attributes separated by commas.</p> <p>Object type: String</p>
<code>baseDN</code>	<p>Use this parameter to specify the base DN for the publishing directory. If <code>dnComps</code> is not set, the server uses the base DN value to start its search in the directory.</p> <p>Example: <code>o=airius.com</code></p> <p>Permissible values: Alphanumeric string up to 255 characters; see "Base Distinguished Name" on page 661.</p> <p>Object type: String</p>

## How Mapping by DN Components Works

Subject names in certificates are in distinguished-name format. A *distinguished name* (DN) uniquely identifies an entry in an LDAP directory; it consists of components that help identify the entry. The following components are commonly used:

- UID, which represents the user ID of a user in the directory
- CN, which represents the common name of a user in the directory
- OU, which represents an organizational unit in the directory
- O, which represents an organization in the directory
- L, which represents a locality in the directory
- ST, which represents a state in the directory
- C, which represents a country in the directory

For example, the following DN represents the user named Jane Doe who works for the sales department at Netscape, which is located in Mountain View in the state of California, United States:

```
CN=Jane Doe, E=jdoe@netscape.com, OU=Sales, O=Netscape,  
L=Mountain View, ST=CA, C=US
```

Certificate Management System uses the components in subject names to construct a DN that it can use as the *base* for searching specific directory entries in order to publish the corresponding certificate information.

For example, suppose the subject name in the certificate is in this form:

```
CN=Jane Doe, OU=Sales, O=Netscape, L=Mountain View, ST=CA,  
C=US
```

Certificate Management System can use some or all of these components (CN, OU, O, L, ST, and C) to build a DN for searching the directory. When configuring the server for LDAP publishing, you can specify which components the server should use to build a DN (that is, components to match attributes in the directory). You do this by configuring the `dnComps` parameter; for details, see Table 17.3 on page 494.



For example, assume you entered components `CN`, `E`, `OU`, `O`, and `C` as values for the `dnComps` parameter. For locating Jane Doe's entry in the directory, Certificate Management System constructs the following DN by reading the DN attribute values from the certificate, and uses the DN as the base for searching the directory:

```
CN=Jane Doe, OU=My Division, O=My Company, C=US
```

Note the following:

- Any components that are not part of the subject name (such as `L`, `ST`, and `E` in this example) are ignored. A subject name does not need to have all of the components that you specify in the configuration.
- Unspecified components are not used to build the distinguished name. In the example, if you did not select the `OU` component in the configuration, Certificate Management System uses this distinguished name as the base for the directory search:

```
CN=Jane Doe, O=My Company, C=US
```

In general, for the `dnComps` parameter, you should enter those DN components that Certificate Management System can use to form the LDAP DN exactly. In certain situations, however, the subject name in a certificate may match more than one entry in the directory. Then, Certificate Management System might not get a single, distinct matching entry from the DN. For example, the subject name

```
CN=Jane Doe, OU=My Division, O=My Company, C=US
```

might match two Jane Does in the directory. If that occurs, Certificate Management System needs additional criteria to determine which entry corresponds to the subject of the certificate.

To specify the components Certificate Management System must use to distinguish between different entries in the directory, use the `filterComps` parameter; for details, see Table 17.3 on page 494.

For example, if you entered `CN`, `OU`, `O`, and `C` as values for the `dnComps` parameter, enter `L` for the `filterComps` parameter only if the `L` attribute can be used to distinguish between entries with identical `CN`, `OU`, `O`, and `C` values.

Consider another example that shows how two directory entries with similar distinguished names can be differentiated by the value of the `UID` attribute: Assume that the two Jane Doe entries are distinguished by the value of the `UID`

attribute. One entry's `UID` value is `janedoe1` and the other entry's `UID` value is `janedoe2`. Because the `UID` attribute corresponds to the `UID` component in a distinguished name, you can set up the subject names of certificates to include the `UID` component.

**Note** By default, the `E`, `L`, and `ST` components are not included in the standard set of certificate request forms provided for end entities. You can add these components to the forms, or you can have the issuing agents insert these components when editing the subject name in the certificate issuance forms.

## Object-Publishing Rules

If you configure Certificate Management System for LDAP publishing, whenever it issues or updates a certificate information, it needs to publish the certificate information in the corresponding directory entry. Object-publishing rules are a set of publishing rules implemented as a Java class and registered in the CMS configuration.

### Built-in Publisher Classes

*Publisher* classes publish an object to an entry in the directory. By default, Certificate Management System provides publisher classes for publishing CA, end-entity, and other certificates and CRLs to entries mapped by the mapper classes. Table 17.4 lists publisher classes provided out of the box.

You can write your own publisher classes by implementing the following Java interface:

```
com.netscape.certsrv.ldappublish.ILdapPublisher
```

For more information on this interface, see the directory named `SDK` on the product CD or check this site for information on CMS SDK:

```
http://home.netscape.com/eng/server/cms
```

Table 17.4 Default publisher classes and their functions

Publisher class name	Description
<code>LdapUserCertPublisher</code>	<p>Publishes or unpublishes a certificate to the <code>userCertificate;binary</code> attribute of the given entry as a DER encoded binary blob.</p> <p>Both the Certificate Manager and Registration Manager provide this plug-in module.</p>
<code>LdapCaCertPublisher</code>	<p>Publishes or unpublishes a certificate to the <code>caCertificate;binary</code> attribute of the given entry. Also converts the object class to a <code>certificateAuthority</code> if it's not one already, and similarly removes the <code>certificateAuthority</code> object class on unpublish if the CA has no other certificates.</p> <p>Only the Certificate Manager provides this plug-in module.</p>
<code>LdapCertAndSubjectPublisher</code>	<p>Publishes or unpublishes a certificate to the <code>userCertificate;binary</code> attribute. The subject name on the certificate is published at the same time to the <code>CertSubjectDN</code> attribute.</p> <p>Only the Certificate Manager provides this plug-in module.</p>
<code>LdapCrlPublisher</code>	<p>Publishes (replaces) a CRL to the <code>certificateRevocationList;binary</code> attribute of the given entry. The entry should be a <code>certificateAuthority</code> object class.</p> <p>Only the Certificate Manager provides this plug-in module.</p>

## Directory Schema Requirements

A directory must be configured with specific attributes and object classes in order to be used for LDAP publishing by Certificate Management System. This section discusses those basic schema requirements.

## Required Schema for Publishing End-Entity Certificates

Certificate Management System publishes an end entity's certificate to the `userCertificate;binary` attribute within the end entity's or subject's directory object. This attribute is multivalued; each value is a DER encoded binary X.509 certificate.

- The LDAP object class `inetOrgPerson` allows this attribute. This object class is supported by Netscape Directory Server versions 1.0, 3.x, and 4.0x.
- The *mix-in* object class `strongAuthenticationUser` allows this attribute and can be combined with any other object class to allow certificate publication to that object.

Certificate Management System does not automatically add this object class in the corresponding Directory Server schema table while publishing or unpublishing end-entity certificates. If the directory object that it finds does not allow the `userCertificate;binary` attribute, the addition or removal of that specific certificate fails.

If you have created user entries as `inetOrgPerson`, the `userCertificate;binary` attribute already exists in the directory. Otherwise, you must add the `userCertificate;binary` attribute to your directory schema table.

## Required Schema for Publishing CA Certificates

Certificate Management System publishes its own CA certificate in the `caCertificate;binary` attribute of the CA's directory object when the server is started; this is the object corresponding to the Certificate Manager's issuer name. This is a required attribute of the object class `certificationAuthority`.

Certificate Management System will add this object class to the directory object for the CA, provided that it finds the CA's directory object.

## Required Schema for Publishing CRLs

Certificate Management System maintains its list of revoked certificates in its internal database; this list is called the certificate revocation list (CRL). You can configure the server to publish the CRL whenever it is generated—which could be when a certificate is revoked or at regular intervals. You can also manually trigger the server to generate a CRL and publish it to the directory. For details, see “Publishing CRLs” on page 523.

The server publishes the updated CRL to the CA's directory object under the attribute `certificateRevocationList;binary`. This attribute is an attribute of the object class `certificationAuthority`. The value of the attribute is the DER encoded binary X.509 certificate revocation list. The CA's entry must already be a certificate authority.

## Directory Synchronization

Certificate Management System and the publishing directory can become out of sync if certificates are issued or revoked while Directory Server is down. Certificates that were issued or revoked need to be published or unpublished manually when Directory Server comes back up.

To help find certificates that are out of sync with the directory—that is, valid certificates that are not in the directory and revoked or expired certificates that are still in the directory—the Certificate Manager or Registration Manager keeps a record of whether a certificate in its internal database has been published to the directory. If Certificate Management System and the publishing directory become out of sync, you can use the Update Directory option in the Certificate Manager Agent Services interface to synchronize the publishing directory with the internal database.

The following choices are available for directory synchronization:

- Search the internal database for certificates that are out of sync and publish or unpublish accordingly.
- Publish certificates that were issued from time A to time B while Directory Server was down. Similarly, unpublish certificates that were revoked or that expired while Directory Server was down.

- Publish or unpublish a range of certificates based on serial numbers (from serial number *xx* to serial number *yy*).

For details, see “Manually Updating Certificate Information in the Directory” on page 520.

# Configuring Subsystems for LDAP Publishing

If you are using an LDAP-compliant directory, such as Netscape Directory Server, to publish and manage your user and group data, you can configure the Certificate Manager or Registration Manager to communicate with this directory. The Certificate Manager can then publish end-entity as well as CA certificates and the certificate revocation list (CRL) to the directory; the Registration Manager can publish end-entity certificates to the directory. This way, your publishing directory acts as a common distribution point for information about users and other entities on the network, including each entity's current security credentials.

The Certificate Manager's or Registration Manager's point of contact with the publishing directory is a single Directory Server. You identify this Directory Server by specifying its host name, port, protocol (either LDAP version 2 or 3) and connection type (LDAP or LDAP over SSL), which the Certificate Manager or Registration Manager stores in its configuration. The specified Directory Server is expected to provide for the necessary distribution requirements to other Directory Servers: by *replication* for directory objects that the specified server manages, and by *referral* for directory objects it does not manage.

This chapter explains how to configure the Certificate Manager or Registration Manager, from the CMS window, to publish certificates to a directory.

The chapter has the following sections:

- Setting Up the Directory for Publishing (page 504)
- Configuring a Certificate Manager for LDAP Publishing (page 507)
- Configuring a Registration Manager for LDAP Publishing (page 515)
- Manually Updating Certificate Information in the Directory (page 520)

## Setting Up the Directory for Publishing

For the Certificate Manager or Registration Manager to publish to your LDAP directory, the directory needs to be set up to receive information from these subsystems. You take the following steps to set up the directory:

- Step 1. Verify the Directory Schema
- Step 2. Add an Entry for the CA
- Step 3. Identify an Entry That Has Write Access
- Step 4. Add Entries for End Entities

### Step 1. Verify the Directory Schema

The directory schema must include all the attributes and object classes explained in “Directory Schema Requirements” on page 499.

### Step 2. Add an Entry for the CA

For the Certificate Manager to publish its CA certificate, the directory must include an entry for the CA. To add this entry in Netscape Directory Server 3.x, use its HTML forms-based interface (the HTTP gateway). In Netscape Directory Server 4.x, you can use the Directory Server window (you can launch this from within Netscape Console). For information on using these interfaces to add an entry for the CA, see the appropriate documentation.



If you are using Netscape Directory Server version 4.x, you can find an online copy of this document at this location:

```
<server_root>/manual/en/slapd/ag/contents.htm
```

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

When adding the entry, be sure to select the entry type based on the distinguished name of the CA:

- If the CA's distinguished name begins with the `CN` component, create a new `Person` entry for the CA. (If you select a different type of entry, the interface may not allow you to specify a value for the `CN` component.)
- If the CA's distinguished name begins with the `OU` component, create a new `Organizational Unit` entry for the CA.

After you select the correct entry type, specify the required information to create the entry. Note that the entry you create doesn't have to be a `certificationAuthority`. Certificate Management System will convert this entry to a `certificationAuthority` automatically by publishing the CA's signing certificate. For details, see "Required Schema for Publishing CA Certificates" on page 500.

**Important** Be sure to carry out this step. If you do not, the CA certificate and CRLs will not be published to the directory.

## Step 3. Identify an Entry That Has Write Access

As part of the process of configuring Certificate Management System to work with Directory Server, you need to specify a distinguished name that has write access to the directory. In other words, to publish certificates and the CRLs to the directory, the Certificate Manager or Registration Manager needs to use a user entry (in the directory) that has *write* access to the directory.

To provide the Certificate Manager or Registration Manager with a user entry that has write permission, do either of the following:

- Give write access to the Certificate Manager's distinguished name.

The Certificate Manager, during its installation, automatically creates a user entry for itself in the directory (for publishing its CA certificate); the entry can be identified by the Certificate Manager's DN:

```
cn=<certificate_authority_name>, o=<org_name>, c=<country>
```

For instructions on giving write access to the Certificate Manager's entry, see your LDAP directory documentation.

- Use the DN of an existing entry that has write access.

You can use the entry of the *Directory Manager* or choose an alternative. In either case, you must identify this user entry to the Certificate Manager and Registration Manager by entering the DN of the user entry as the `Bind` as parameter value when you configure the subsystems for LDAP publishing. See "Identifying a Certificate Manager's Publishing Directory" on page 508.

You also need to know the password for the DN with write access, because you will be required to enter it in the Certificate Manager's or Registration Manager's configuration.

Once the Certificate Manager or Registration Manager is configured to publish to the directory, the following operations are performed automatically:

- When the Certificate Manager starts up, it publishes its CA certificate to the configured directory.
- When the Certificate Manager or Registration Manager issues a new certificate, it publishes the certificate to the configured directory.
- When the Certificate Manager revokes a certificate, it removes that certificate from the configured directory.
- When a certificate expires, the Certificate Manager or Registration Manager can remove that certificate from the configured directory. The server uses the Job Scheduler component to accomplish this task; see "Directory Update and Notification" on page 358.

- When the certificate revocation list is created or updated (either through the CMS window or through a certificate revocation in the agent or end-entity interface), the Certificate Manager publishes that list to the configured directory.

## Step 4. Add Entries for End Entities

You need to add an entry for each end entity for whom you want a certificate published. If the end entity does not have an entry in the directory, the Certificate Manager or Registration Manager will not be able to publish the end entity's certificate.

You can use the tools provided with Directory Server to add an entry for each end entity. These entries must belong to an object class, such as `inetOrgPerson`, that allows the `userCertificate;binary` attribute. For details, see “Required Schema for Publishing End-Entity Certificates” on page 500.

**Note** If you configured Certificate Management System to use directory-based authentication for end entities and are using the same directory for authentication and LDAP publishing, you may not have to deal with this issue. Certificate Management System will not issue certificates to end entities that do not have entries in the directory. See “End-Entity Authentication During Certificate Enrollment” on page 265.

# Configuring a Certificate Manager for LDAP Publishing

This section explains how to identify the publishing directory and configure the Certificate Manager to publish certificates to that directory. For information on configuring a Certificate Manager for publishing CRLs, see “Publishing CRLs” on page 523.

Configuring a Certificate Manager for LDAP publishing involves the following tasks:

- Identifying a Certificate Manager's Publishing Directory
- Configuring Mapper and Publisher Classes for the CA Certificate
- Configuring Mapper and Publisher Classes for End-Entity Certificates

**Note** Before you configure a Certificate Manager for LDAP publishing, make sure that the directory you intend to use has been configured properly. See “Setting Up the Directory for Publishing” on page 504.

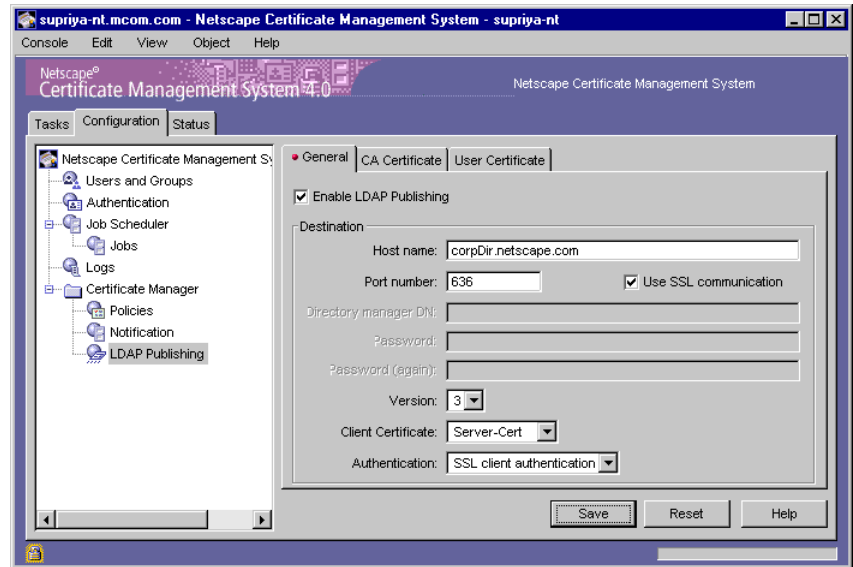
## Identifying a Certificate Manager's Publishing Directory

To identify the Directory Server instance that a Certificate Manager should use for publishing the CA certificate, end-entity certificates, and CRLs:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.

3. In the navigation tree, click Certificate Manager, then click LDAP Publishing.

The right pane shows the publishing details necessary for the subsystem to publish to an LDAP-compliant directory.



4. To enable LDAP publishing, check the Enable LDAP Publishing option.
5. In the Destination section, identify a Directory Server instance.

**Host name.** Type the full host name of the Directory Server instance. The Certificate Manager uses this name to locate the directory. The format for the host name must be as follows:

`<machine_name>.<your_domain>.<domain>`

**Port number.** Type the TCP/IP port number at which this Directory Server is listening to publishing requests from the Certificate Manager. The port you specify should be unique on the Directory Server host system; make sure no other application is attempting to use the same port.

**Authentication.** Select the authentication type. The choices are “Basic authentication” and “SSL client authentication.” If you select “Basic authentication,” you must specify the Bind as parameter. If you select “SSL

client authentication,” you must check the “Use SSL communication” box and identify the certificate that the Certificate Manager must use for SSL client authentication to the directory.

**Use SSL communication.** Check this box if the port number you typed is an SSL port; uncheck the box if the port is non-SSL. The port type you specify determines whether the Certificate Manager needs to do SSL client authentication prior to publishing certificates and CRLs to the directory. Also, before checking this box, make sure that the specified Directory Server is configured for SSL-enabled communication.

**Client certificate.** Select the certificate you want the Certificate Manager to use for SSL client authentication to the publishing directory. By default, the Certificate Manager uses its SSL server certificate for this purpose (see “SSL Server Key Pair and Certificate” on page 186).

**Directory manager DN.** Type the distinguished name (DN) of an entry in your LDAP directory that has *write* permission to the CA and end-entity entries in the directory. You specified this when setting up the directory for publishing; see “Step 3. Identify an Entry That Has Write Access” on page 505. The Certificate Manager uses this DN to access the directory tree and to publish to the directory. The access control set up for this DN determines whether the Certificate Manager can perform publishing. Typically, you would want to enter the directory manager’s DN because it has read-write permission to the entire directory tree (the root DN). For more information on root DN, see “Root Distinguished Name” on page 661.

**Password.** Type the password for this DN. If you change the password, the server updates the single sign-on password cache with the new password; for details, see “Required Start-up Information” on page 106.

**Confirm password.** Retype the password exactly as you typed it in the previous field.

**Version.** Select the LDAP protocol version. The choices are version 2 and version 3. If the directory you want the Certificate Manager to publish to is based on Netscape Directory Server 1.x, select version 2. For Directory Server versions 3.x and later, select LDAP version 3.

6. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

## Configuring Mapper and Publisher Classes for the CA Certificate

You can configure the Certificate Manager to publish its CA certificate to the directory. If so configured, the Certificate Manager publishes the certificate to the directory specified in the General tab; see “Identifying a Certificate Manager’s Publishing Directory” on page 508.

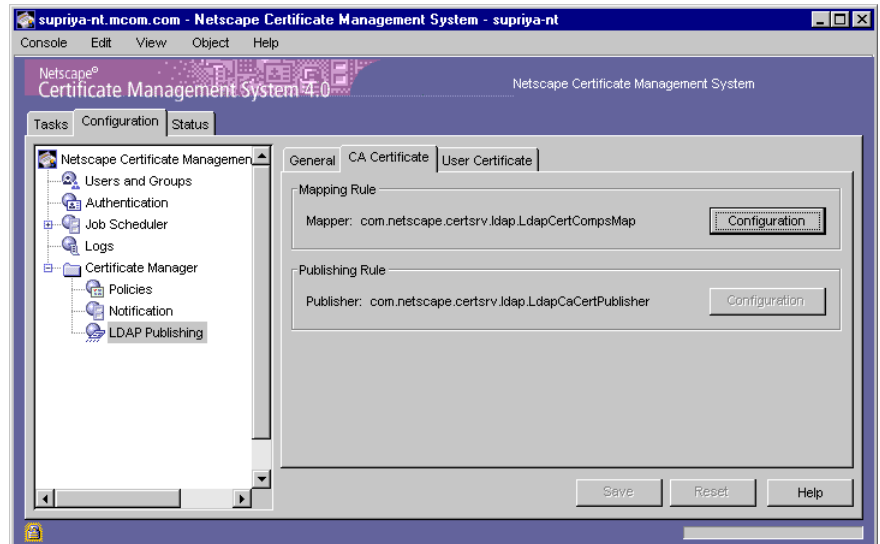
Only a Certificate Manager can publish CA certificates; a Registration Manager cannot publish CA certificates.

To specify the details necessary for publishing the CA certificate to a directory:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, click Certificate Manager, then click LDAP Publishing.

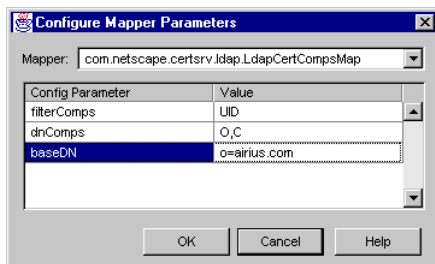
The right pane shows the publishing details necessary for the Certificate Manager to publish to an LDAP-compliant directory.

4. Click the CA Certificate tab.



5. In the CA Certificate tab, click Configuration.

The Configure Mapper Parameters window appears.



6. From the Mapper drop-down list, select the mapper class you want the Certificate Manager to use for locating the CA entry in the publishing directory.

By default, the list box shows the default mapper class, called `LdapCertCompsMap`, provided by the Certificate Manager. If you have registered any custom mapper classes, they too are available for selection.

7. Enter the appropriate values for the configuration parameters that define the mapper class you chose.

For detailed information about the default mapper class, see “Built-in Mapper Classes” on page 491.

8. Click OK.

You are returned to the CA Certificate tab.

9. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.



## Configuring Mapper and Publisher Classes for End-Entity Certificates

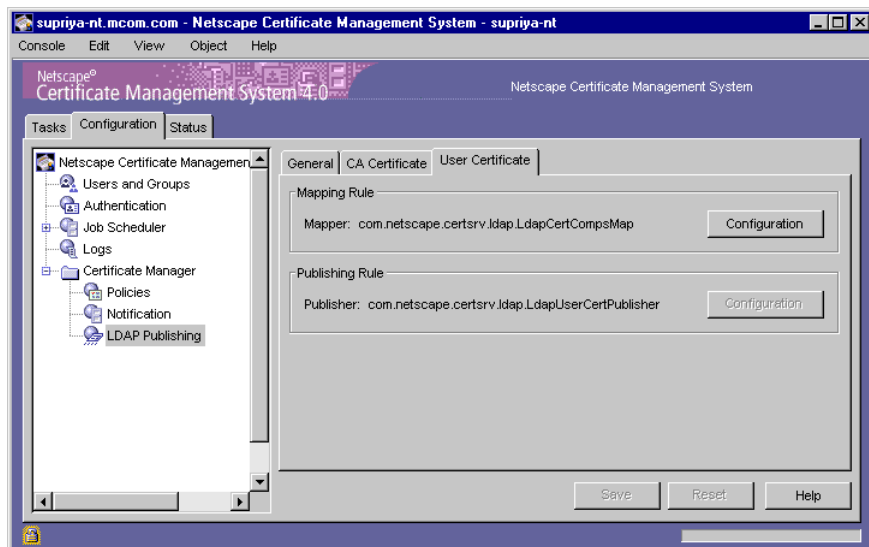
You can configure a Certificate Manager to publish end-entity certificates to an LDAP-compliant directory. If so configured, the subsystem publishes end-entity certificates to the directory specified in the General tab; see “Identifying a Certificate Manager’s Publishing Directory” on page 508.

To specify the details necessary for publishing end-entity certificates to a directory:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, click Certificate Manager, then click LDAP Publishing.

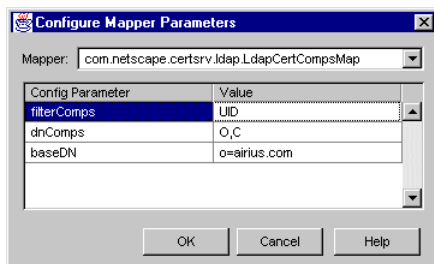
The right pane shows the publishing details necessary for the subsystem to publish to an LDAP-compliant directory.

4. Click the User Certificate tab.



5. In the User Certificate tab, click Configuration.

The Configure Mapper Parameters window appears.



6. From the Mapper drop-down list, select the mapper class you want the Certificate Manager to use for locating end-entity entries in the publishing directory.

By default, the list box shows the default mapper class, called `LdapCertCompsMap`, provided by the Certificate Manager. If you have registered any custom mapper classes, they too are available for selection.

7. Enter the appropriate values for the configuration parameters that define the mapper class you chose.

For detailed information about the default mapper class, see “Built-in Mapper Classes” on page 491.

8. Click OK.

You are returned to the User Certificate tab.

9. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

# Configuring a Registration Manager for LDAP Publishing

Configuring a Registration Manager for LDAP publishing involves the following tasks:

- Identifying a Registration Manager's Publishing Directory
- Configuring Mapper and Publisher Classes for End-Entity Certificates

**Note** Before you configure a Registration Manager for LDAP publishing, make sure that the directory you intend to use has been configured properly. See “Setting Up the Directory for Publishing” on page 504.

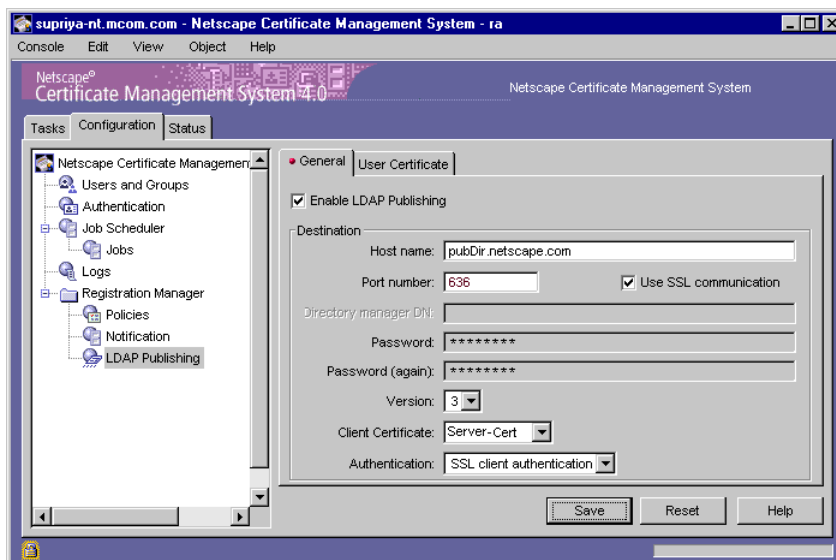
## Identifying a Registration Manager's Publishing Directory

To identify the Directory Server instance that a Registration Manager should use for publishing end-entity certificates:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.

3. In the navigation tree, click Registration Manager, then click LDAP Publishing.

The right pane shows the publishing details necessary for the subsystem to publish to an LDAP-compliant directory.



4. To enable LDAP publishing, check the Enable LDAP Publishing option.
5. In the Destination section, identify a Directory Server instance:

**Host name.** Type the full host name of the Directory Server instance. The Registration Manager uses this name to locate the directory. The format for the host name must be as follows:

<machine\_name>.<your\_domain>.<domain>

**Port number.** Type the TCP/IP port number at which this Directory Server is listening to publishing requests from the Registration Manager. The port you specify should be unique on the Directory Server host system; make sure that no other application is attempting to use the same port.

**Authentication.** Select the authentication type. The choices are “Basic authentication” and “SSL client authentication.” If you select “Basic authentication,” you must specify the Bind as parameter. If you select “SSL

client authentication,” you must check the “Use SSL communication” box and identify the certificate that the Registration Manager must use for SSL client authentication to the directory.

**Use SSL communication.** Check this box if the port number you typed is an SSL port; uncheck the box if the port is non-SSL. The port type you specify determines whether the Registration Manager needs to do SSL client authentication prior to publishing certificates to the directory. Also, before checking this box, make sure that the specified Directory Server is configured for SSL-enabled communication.

**Client certificate.** Select the certificate you want the Registration Manager to use for SSL client authentication to the publishing directory. By default, the Registration Manager uses its SSL server certificate for this purpose (see “SSL Server Key Pair and Certificate” on page 188).

**Directory manager DN.** Type the distinguished name (DN) of an entry in your LDAP directory that has *write* permission to the end-entity entries in the directory. You specified this when setting up the directory for publishing; see “Step 3. Identify an Entry That Has Write Access” on page 505. The Registration Manager uses this DN to access the directory tree and publish to the directory. The access control set up for this DN determines whether the Registration Manager can perform publishing. Typically, you would want to enter the directory manager’s DN because it has read-write permission to the entire directory tree (the root DN). For more information on root DN, see “Root Distinguished Name” on page 661.

**Password.** Type the password for this DN. If you change the password, the server updates the single sign-on password cache with the new password; for details, see “Required Start-up Information” on page 106.

**Confirm password.** Retype the password exactly as you typed it in the previous field.

**Version.** Select the LDAP protocol version. The choices are version 2 and version 3. If the directory you want the Registration Manager to publish to is based on Netscape Directory Server 1.x, select version 2. For Directory Server versions 3.x and later, select LDAP version 3.

6. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

## Configuring Mapper and Publisher Classes for End-Entity Certificates

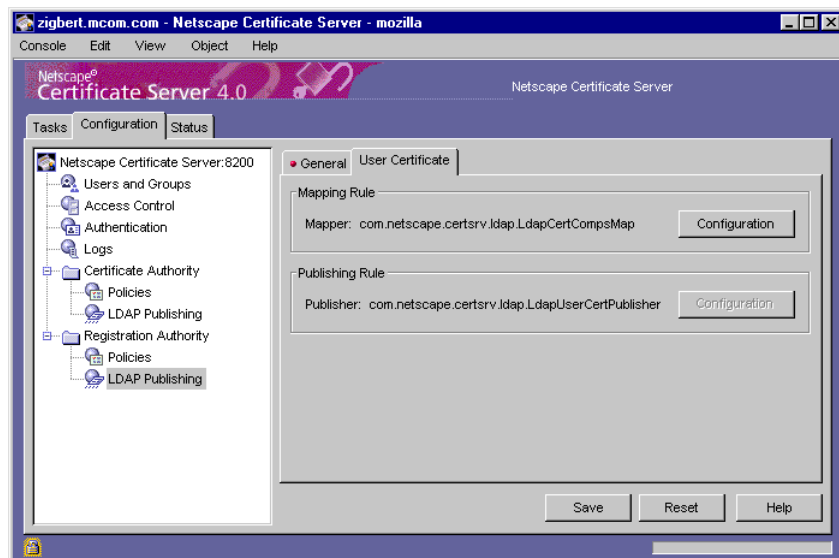
You can configure the Registration Manager to use specific object mapping and publishing rules (classes) to locate end-entity entries in the publishing directory to publish or update certificate information.

To specify the details necessary for publishing end-entity certificates to a directory:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, click Registration Manager, then click LDAP Publishing.

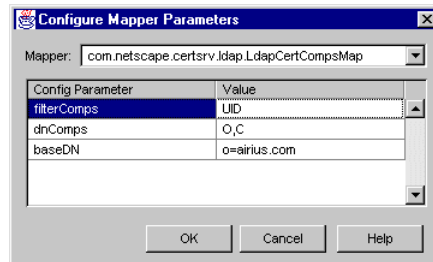
The right pane shows the publishing details necessary for the subsystem to publish to an LDAP-compliant directory.

4. Click the User Certificate tab.



5. In the User Certificate tab, click Configuration.

The Configure Mapper Parameters window appears.



6. From the Mapper drop-down list, select the mapper class you want the Registration Manager to use for locating end-entity entries in the publishing directory.

By default, the list box shows the default mapper class, called `LdapCertCompsMap`, provided by the Registration Manager. If you have registered any custom mapper classes, they too are available for selection.

7. Enter the appropriate values for the configuration parameters that define the mapper class you chose.

For detailed information about the default mapper class, see “Built-in Mapper Classes” on page 491.

8. Click OK.

You are returned to the User Certificate tab.

9. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

# Manually Updating Certificate Information in the Directory

Normally you do not need to manually update the directory with certificate-related information; If configured properly, the Certificate Manager handles most of the updates automatically. However, a situation might arise in which you need to update the directory manually. For example, Directory Server might be down for a time and be unable to receive changes from the Certificate Manager.

In such a situation, you should use the Update Directory Server form in the Certificate Manager Agent Services interface to manually update the directory with certificate-related information. This form lets you initiate a combination of the following operations:

- Update the directory with certificates.
- Remove expired certificates from the directory. Note that you can automate removal of expired certificates from the publishing directory by scheduling a CMS job. For details, see “Directory Update and Notification” on page 358.
- Remove revoked certificates from the directory.

A Certificate Manager’s publishing directory can be manually updated by a Certificate Manager agent only. Agent operations are restricted to those with a proper agent certificate; see “Agent’s Certificate for SSL Client Authentication” on page 137.

**Note** A Registration Manager’s publishing directory cannot be updated.

For complete details on agent operations, see *Netscape Certificate Management System Agent’s Guide*.

To manually update the directory with changes:

1. Go to the Certificate Manager Agent Services page.

You must submit the proper client certificate to get access to this page.

2. Click the Update Directory Server link.

The Update Directory Server page appears.



3. Select the appropriate options.
4. When you are done specifying the changes that you want updated, click Update Directory.

The subsystem starts updating the directory with the certificate information in its internal database. In some circumstances—for example, if the changes are substantial—updating the directory can take considerable time. During this period, any changes made through the subsystem (for example, any new certificates issued or any certificates revoked) may not be included in the update. If you have issued or revoked any certificates during that time, you need to update the directory again to reflect those changes.

When the directory update is complete, the subsystem displays a status report. If for some reason the process gets interrupted, the subsystem logs an appropriate error message. Be sure to check logs if that happens; for details, see “Monitoring Logs” on page 586.



## Publishing CRLs

Server and client applications that use public-key certificates as tokens of identification need access to information about the validity of a certificate; they need to know whether the certificate has been revoked. One of the standard, secure mechanisms for conveying this information is by publishing a list of revoked certificates. This list is known as the *Certificate Revocation List* (CRL).

The CRL is a publicly available list of certificates that have been revoked. This list ensures that revoked certificates are not misused.

This chapter explains certificate revocation in general and explains how to configure Netscape Certificate Management System (CMS) to publish CRLs.

The chapter has the following sections:

- CRL Authorities (page 524)
- CRL Issuing Points (page 524)
- Reasons for Revoking a Certificate (page 525)
- Updating CRLs Automatically (page 526)
- Updating CRLs Manually (page 529)

## CRL Authorities

CRLs are issued and digitally signed by the certificate authority (CA) that issued the certificates included in the revocation list. The CA's function includes creating the CRLs periodically and distributing them to other applications by appropriate means. For example, the CA may publish the CRL to a global directory which other applications may use for checking the revocation status of a certificate or from which other applications can retrieve the CRL.

In Certificate Management System, the Certificate Manager creates the CRL and publishes it to an LDAP-compliant directory, if configured to do so. A Registration Manager cannot create or publish CRLs.

CRLs can be made available to the PKI entities through a number of mechanisms. This document focuses on two of the most common mechanisms: publishing CRLs in the LDAP directory and retrieving CRLs over HTTP.

## CRL Issuing Points

Because CRLs can grow very large, several mechanisms were developed to minimize overhead of retrieving and delivering large CRLs. One of these mechanism is based on partitioning the entire certificate space and associating a separate CRL with every partition. This partition is called a *CRL issuing or distribution point*—it is the location where a subset of all the revoked certificates are maintained. Partitioning can be based on revocation reason, on whether the revoked certificate is a CA certificate or end-entity certificate, on end users' names, and so on. Each issuing point is identified by a set of names, which can be in various forms.

Once the issuing points have been defined, they can be included in certificates so that an application that needs to check the revocation status of a certificate can access the CRL issuing points specified in the certificate instead of the master or main CRL—the application would check the CRL maintained at the issuing point, which would be smaller in size compared to the master CRL, and thus speed up the revocation-status-checking process.

CRL distribution points can be associated with certificates using the `CRLDistributionPoint` extension in the certificates.

By default, Certificate Management System generates and publishes a single CRL called the *master CRL*. This is explained in “Updating CRLs Automatically” on page 526. However, you can define CRL issuing points that contain a subset of the revoked certificates included in the master CRL. For details on configuring Certificate Management System to publish CRLs to several issuing points, check this site:

<http://home.netscape.com/eng/server/cms>

To enable you to add the CRL issuing points in certificates, Certificate Management System provides a policy plug-in module that is based on the `CRLDistributionPoint` extension. You can configure the policy module to add the required issuing points in the certificates the server issues. For details, see “CRL Distribution Point Extension Policy” on page 438.

## Reasons for Revoking a Certificate

A Certificate Manager (the CA) can revoke any certificate it has issued. A certificate needs to be revoked if one or more of the following situations occur:

- The owner of the certificate has changed status and no longer has the right to use the certificate.
- The private key of a certificate owner has been compromised.
- The certificate owner doesn't want to use the certificate.
- The private key of the CA that issued the certificate has been compromised.

Whenever a certificate is revoked (by administrators, agents, or end entities), the CA keeps track of revoked certificates in a CRL. By revoking a certificate, you are notifying other users that the certificate is no longer valid. You make this notification by publishing a list of the revoked certificates. This list is called the CRL.

At the time of this writing, the current versions of Netscape products other than Certificate Management System do not have the ability to automatically check to see whether a certificate has been revoked. However, Netscape clients do give the user the ability to check this if the certificate includes the `NetscapeRevocationURL` extension.

In addition, from the end entity services interface, Netscape client users can manually check the revocation status of a particular certificate and automatically import the latest version of the CRL into their browsers. If your users are not using Netscape clients, they can download the latest CRL in binary form to a local file, and then import this file into their browsers by an appropriate method. Users can also view the CRL header information of the master or full CRL published by Certificate Management System, which contains the date and time of the latest update, and then compare this information to that in their browser's CRL to see if they have the latest version.

Because Netscape servers currently cannot check the revocation status of a certificate, you should use other forms of access control. For example, you can remove individual users from access groups to prevent them from accessing the server.

Because Certificate Management System can check the revocation status of the certificates that it issues, you do not need to rely on other forms of access control.

## Updating CRLs Automatically

Normally, when you revoke a certificate, the Certificate Manager automatically updates the status of the certificate in its internal database. In addition, the Certificate Manager also maintains a CRL in its internal database. You can configure the server to generate the CRL either every time a certificate is revoked or at a periodic interval.

You can also configure the Certificate Manager to publish the CRL it maintains to a publishing directory, for example, to the directory in which end-entity certificates are published. If you do so, the server attempts to publish to the directory whenever it generates a new CRL—which could be either every time a certificate is revoked or at a periodic interval.

Here are a few things about the CRLs that a Certificate Manager can generate and publish:

- By default, the server generates the CRL whenever a certificate is revoked. Because CRLs are published whenever they are generated, the server attempts to publish the CRL to the publishing directory every time it revokes

a certificate. You can configure the server to generate the CRL at a regular interval instead, say every day or week, and thus change the publishing interval.

- The server publishes the CRL to the CA's entry in the directory, replacing the old CRL with the new one; the old CRL is not saved. For details about the CA's entry, see "Required Schema for Publishing CRLs" on page 501.
- The server can generate and publish CRLs conforming to either X.509 version 1 or version 2 standards; you can configure it for either one by enabling or disabling the CRL extension-specific options in the server's configuration. The default configuration is to publish version 1 CRLs.
- The server supports some CRL extensions. These are listed in Table 19.1.

Table 19.1 CRL extensions and CRL entry extensions supported by the Certificate Manager

Component	Supported	Not supported
CRL extensions	<ul style="list-style-type: none"><li>• Authority key identifier</li><li>• CRL number</li><li>• Issuing distribution point</li></ul>	<ul style="list-style-type: none"><li>• Issuer alternative name</li><li>• Delta CRL indicator</li><li>• Certificate Issuer</li></ul>
CRL Entry extensions	<ul style="list-style-type: none"><li>• Reason code</li><li>• Invalidity date</li></ul>	<ul style="list-style-type: none"><li>• Hold instruction code</li></ul>

## Configuring a Certificate Manager for Publishing CRLs

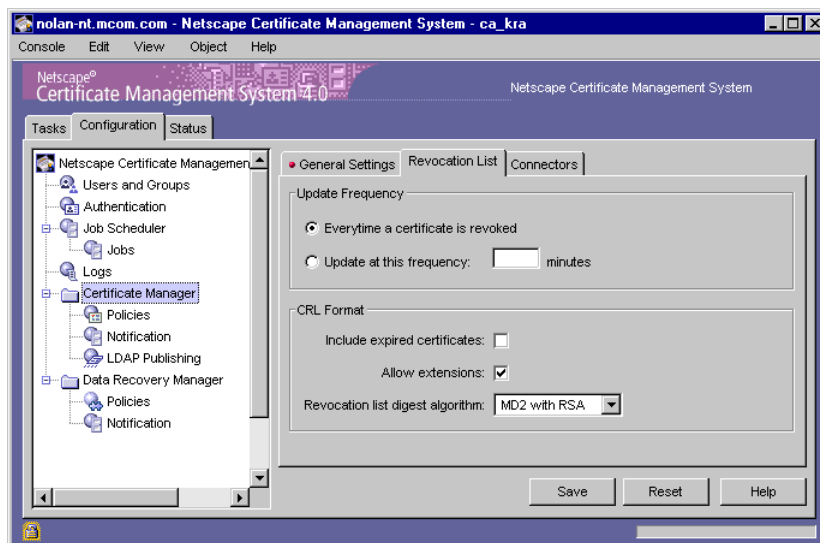
You can configure the Certificate Manager to publish CRLs to an LDAP-compliant directory currently configured for publishing end-entity certificates; see "Identifying a Certificate Manager's Publishing Directory" on page 508). The Certificate Manager publishes the CRL to the specified directory using the base DN and password you specified. To locate the CA's entry in the directory, the server uses the configuration specified for publishing the CA certificate; see "Configuring Mapper and Publisher Classes for the CA Certificate" on page 511.

As a part of this configuration, you can specify information, such as the publishing interval, whether to include extensions, and the message digest algorithm the Certificate Manager should use for signing the CRL object.

To configure a Certificate Manager to publish the CRL to the directory:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, click Certificate Manager.

The General Settings tab appears.



4. Click the Revocation List tab.
5. In the Update Frequency section, specify the interval for publishing the CRL to the directory:

**Every time a certificate is revoked.** Select this option if you want the Certificate Manager to generate the CRL every time it revokes a certificate. Keep in mind that the server attempts to publish the CRL to the configured directory whenever it is generated, in this case, every time a certificate is revoked. Publishing a CRL can be time consuming if the CRL is large. Configuring the Certificate Manager to publish CRLs every time a certificate is revoked may engage the server for a considerable amount of time; during this time, the server will not be able to service any requests it receives or update the directory with any changes it receives.



**Update at this frequency.** Select this option if you want the Certificate Manager to generate the CRL at regular intervals. In this case, the server publishes the CRL to the configured directory at configured intervals.

In the adjoining text field, type the interval, in minutes, at which the Certificate Manager should publish CRLs. For example, if you want the server to publish CRLs every day, you should type 1440 (minutes) in this field.

6. In the CRL Format section, specify the format for publishing the CRL:

**Include expired certificates.** Check this box if you want the server to publish expired certificates to the publishing directory.

**Allow extensions.** Check this box if you want to allow extensions in CRLs. If you enable this option, the server generates and publishes CRLs conforming to X.509 version 2 standard. If you disable this option, the server generates and publishes CRLs conforming to X.509 version 1 standard. By default, the server publishes version 1 CRLs.

**Revocation list digest algorithm.** Select the algorithm the server should use to sign the CRL for publishing. Available choices are MD2 with RSA, MD5 with RSA, SHA1 with RSA, and SHA1 with DSA.

7. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

## Updating CRLs Manually

Normally you do not need to manually update the directory with CRL-related information. If configured properly, the Certificate Manager handles most of the updates automatically. However, a situation may arise in which you need to update the directory manually. For example, Directory Server may be down for a time and thus unable to receive changes from the Certificate Manager.

In such a situation, you should use the Update Certificate Revocation List form in the Certificate Manager Agent Services interface to manually update the directory with CRL-related information.

Keep in mind that a Certificate Manager's publishing directory can be updated by a Certificate Manager agent only. Agent operations are restricted to those with a proper agent certificate.

To manually update the directory with changes:

1. Go to the Certificate Manager Agent Services page. You must submit the proper client certificate to get access to this page.
2. Click Update Revocation List.

The Update Certificate Revocation List page appears.

3. From the Signature algorithm drop-down list, select the appropriate signature algorithm. Choices available are: MD5 with RSA Encryption and SHA-1 with RSA Encryption.
4. Click Update.

The Certificate Manager starts updating the directory with the CRL in its internal database. In some circumstances, for example, if the CRL is large, updating the directory may take considerable time. During this period, any changes made through Certificate Management System (for example, any new certificates issued or any certificates revoked) may not be included in the update. If you have issued or revoked any certificates during that time, you need to update the directory with those changes.

When the directory is updated, the Certificate Manager will display a status report. If the process gets interrupted for some reason, the subsystem logs an appropriate error message. Be sure to check logs if that happens; see "Monitoring Logs" on page 586.

# 8

## *Agent and End-Entity Interfaces*

*Chapter 20* Introduction to End-Entity and Agent Interfaces

*Chapter 21* Customizing End-Entity and Agent Interfaces



# Introduction to End-Entity and Agent Interfaces

Netscape Certificate Management System (CMS) provides HTML forms-based interfaces for agents and end entities to use in performing certificate- and key-related operations. This chapter introduces these forms and explains how they work. You can use the forms as they are provided out of the box or customize them to meet your organization's requirements.

This chapter has the following sections:

- End-Entity Services (page 533)
- Configuring End-Entity Interaction with Subsystems (page 537)
- Agent Services (page 541)

For details on customizing these forms, see “Customizing End-Entity and Agent Interfaces” on page 547.

## End-Entity Services

Certificate Management System provides HTML forms for the various entities—people, routers, servers, and others—that use certificates to identify themselves and that need to be able to request certificate issuance and management operations. These forms, collectively called the *end-entity services* interface, use different protocols and life-cycle management procedures for different kinds of

end entities. For example, the Certificate Manager provides separate certificate enrollment forms for clients such as Netscape Navigator 3.x, versions of Netscape Communicator later than 4.5, and Microsoft Internet Explorer. The reason for this is that end entities running Navigator 3.x and Communicator versions earlier than 4.5 present an enrollment form based on the use of the HTML tag `KEYGEN` to generate keys; end entities running Internet Explorer present a form based on PKCS #10, the RSA standard for certificate request syntax.

Figure 20.1 shows the end-entity services interface hosted by a Registration Manager.

Figure 20.1 End-entity services interface

**Registration Manager**

**Enrollment** | Renewal | Revocation | Retrieval

**Directory And PIN Based User Enrollment**

Use this form to submit a request for a personal certificate based on information supplied by your organization's directory. Your user ID and password for the directory and a one time personal identification number (PIN) assigned by your system administrator are required for this automatic method of certificate issuance. If the user ID, password and PIN are correct your certificate will be issued automatically.

**Important:** Be sure to request your certificate on the same computer on which you plan to use the certificate.

**User's Identity**  
Enter your user ID and password for your organization's directory and the one time PIN given by your system administrator. This information will be used to verify your identity and to obtain information from the directory to fill in the certificate.

User Id:

Password:

Enter the PIN your system administrator has communicated to you for certificate enrollment.

PIN:

**Key Length Information**  
When you submit this form, the browser generates a private key and a public key. It retains the private key and submits the public key along with your request for a

For a summary of the various end entities, protocols, cryptographic algorithms, and key pairs (single or dual) supported by Certificate Management System, see Table 20.1 on page 536.

For a complete list of the end-entity forms—for enrollment, renewal, retrieval, revocation, and key recovery—that come with Certificate Management System, see “Summary of End-Entity Forms and Templates” on page 553.

## How Client Type Determines the End-Entity Interface

Each type of end-entity form provided by Certificate Management System is served by a servlet. This servlet determines which version of the form to present based on information about the end entity (the type, version, language, and so on), information in the form itself, and other factors.

Each form also specifies both an authentication manager and an output template:

- An authentication manager is a configured instance of an authentication plug-in module. When Certificate Management System receives a request from an end entity, it uses the authentication manager specified by the request to determine how to authenticate the end entity. For more information, see “Adding an Authentication Instance” on page 317.
- The output template is an HTML page with embedded JavaScript used to return information from the end entity to the servlet. For more information, see “Responses and Output Templates” on page 550.

Based on all the information, a form's servlet sends the end entity the version of the form (including the embedded JavaScript code) appropriate for that end entity. For example, in the case of end entities that support the `KEYGEN` tag, the Certificate Manager or Registration Manager sends a form that uses `KEYGEN` to generate keys and formulate a certificate request. In the case of end entities that support the Certificate Management Message Format (CMMF) protocol, the Certificate Manager or Registration Manager sends a form that uses a JavaScript API to fully automate both key generation and certificate issuance.

# Certificate Request Formats Specific to End Entities

Table 20.1 lists the forms provided by the Certificate Manager and Registration Manager for certificate issuance and life-cycle management operations, and indicates supported authentication mechanisms and request formats.

You can customize any of the default forms and their corresponding servlets and output templates. For details, see “Customizing End-Entity and Agent Interfaces” on page 547.

Table 20.1 Summary of end-entity forms, authentication mechanisms and certificate request formats

Form for end-entity operation	Authentication mechanism	Supported certificate request formats
<b>Certificate enrollment</b>		
Client (end user) certificates	Manual or directory based	<ul style="list-style-type: none"><li>• KEYGEN for Navigator/Communicator</li><li>• PKCS #10 for Internet Explorer</li><li>• Certificate Request Message Format (CRMF) for future versions of Communicator</li></ul>
Server certificates	Manual or directory based	PKCS #10
Cisco routers	Manual	Certificate Enrollment protocol (CEP)
<b>Certificate renewal</b>		
Client (end user) certificates	SSL client authentication	<ul style="list-style-type: none"><li>• KEYGEN for Navigator/Communicator</li><li>• PKCS #10 for Internet Explorer</li><li>• CRMF for future versions of Communicator</li></ul>
Server certificates	Manual	PKCS #10
Cisco routers	Manual	CEP



Table 20.1 Summary of end-entity forms, authentication mechanisms and certificate request formats (Continued)

Form for end-entity operation	Authentication mechanism	Supported certificate request formats
<b>Certificate revocation</b>		
Client (end user) certificates	SSL client authentication	<ul style="list-style-type: none"><li>• KEYGEN for Navigator/Communicator</li><li>• PKCS #10 for IE</li><li>• CRMF for future versions of Communicator</li></ul>
Server certificates	Manual	PKCS #10
Cisco routers	Manual	CEP
<b>Encryption private key storage and recovery</b>		
Client (end user) certificates	Not applicable	<ul style="list-style-type: none"><li>• Not supported for clients that can't generate dual key pairs</li><li>• CRMF for future versions of Communicator</li></ul>

# Configuring End-Entity Interaction with Subsystems

You can configure end-entity interaction with a Certificate Manager or a Registration Manager, or with both. End entities cannot interact with a Data Recovery Manager directly; they must interact through a Certificate Manager or Registration Manager.

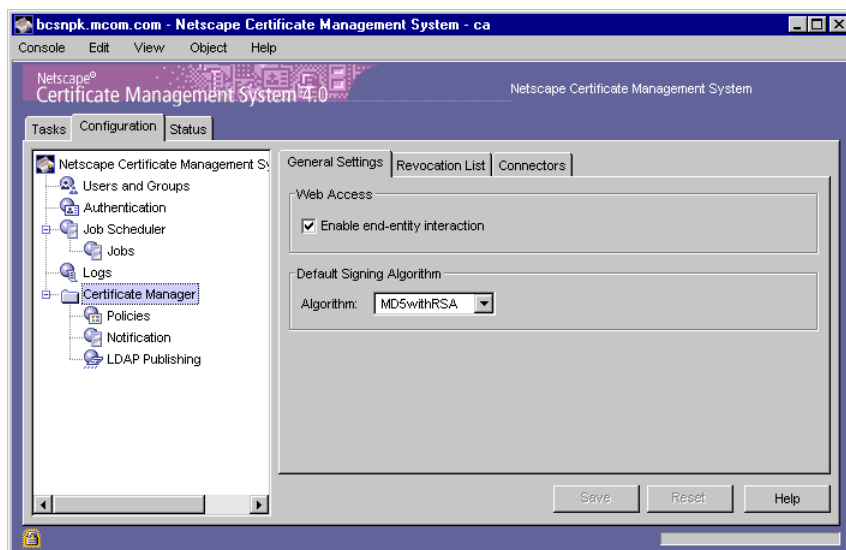
By default, the Certificate Manager is configured for end-entity interaction; the Registration Manager is not configured for end-entity interaction.

## Enabling End-Entity Interaction with a Certificate Manager

To enable end-entity interaction with a Certificate Manager:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, click Certificate Manager.

The General Setting tab appears.



4. In the Web Access section, check the “Enable end-entity interaction” option if you want end entities to be able to interact with the selected Certificate Manager via the HTTPS port; leave it unchecked to disable end-entity interaction with the server. Note that if you disable end-entity interaction, the Network tab still shows the HTTPS port and allows you to configure it (see “Configuring Port Numbers” on page 125). However, you should know that the server ignores this port.

5. In the Default Signing Algorithm section, select the signing algorithm the Certificate Manager should use for signing certificates. The choices are “MD2 with RSA,” “MD5 with RSA,” and “SHA1 with RSA,” if the CA’s signing key type is RSA and “SHA1 with DSA,” if the CA’s signing key type is DSA. Note that the signing algorithm specified in the Certificate Manager’s policy configuration overrides the algorithm you select here. For information on a Certificate Manager’s policy configuration, see “Policies” on page 399.
6. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

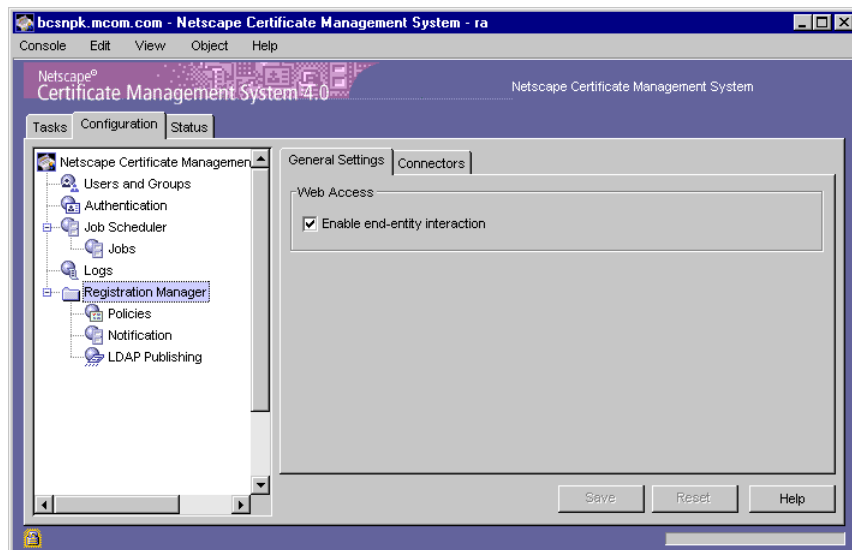
## Enabling End-Entity Interaction with a Registration Manager

To enable end-entity interaction with a Registration Manager:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.

3. In the navigation tree, click Registration Manager.

The General Setting tab appears.



4. In the Web Access section, check the “Enable end-entity interaction” option if you want end entities to be able to interact with the selected Registration Manager via the HTTPS port; leave it unchecked to disable end-entity interaction with the server. Note that if you disable end-entity interaction, the Network tab still shows the HTTPS port and allows you to configure it (see “Configuring Port Numbers” on page 125). However, you should know that the server ignores this port.
5. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

# Agent Services

As an administrator, you can designate privileged users, called agents, for each subsystem. Agents are responsible for the day-to-day operation of requests from end entities. To enable agents to accomplish their duties, Certificate Management System provides a set of HTML forms for Certificate Manager, Registration Manager, and Data Recovery Manager agents. Collectively, these forms are called the *Agent Services* interface.

Depending on the choices you made during installation, a combination of the following agent services will be installed:

- Certificate Manager Agent Services
- Registration Manager Agent Services
- Data Recovery Manager Agent Services

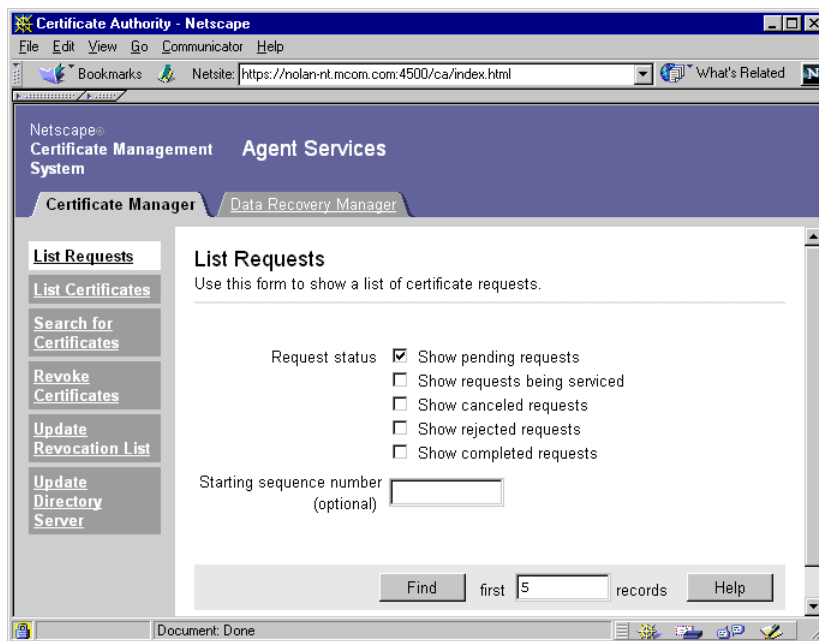
This section gives an overview of these forms and explains how to access them. For a complete list of the agent forms and output templates that come with Certificate Management System, see “Summary of Agent Forms and Templates” on page 563. For step-by-step instructions on using the agent forms, see *Netscape Certificate Management System Agent's Guide*. For information on locating this guide, see “Where to Go for Related Information” on page 27.

Note that accessing the Agent Services interface is a privileged operation, requiring certificate-based (or *strong*) authentication. It can be done only by users belonging to authorized agent groups maintained by Certificate Management System in its internal database. For details, see “Agents” on page 135.

## Certificate Manager Agent Services

The Certificate Manager Agent Services interface enables a Certificate Manager agent to interact with the Certificate Manager (the server). Figure 20.2 shows the Certificate Manager Agent Services interface.

Figure 20.2 Certificate Manager Agent Services interface



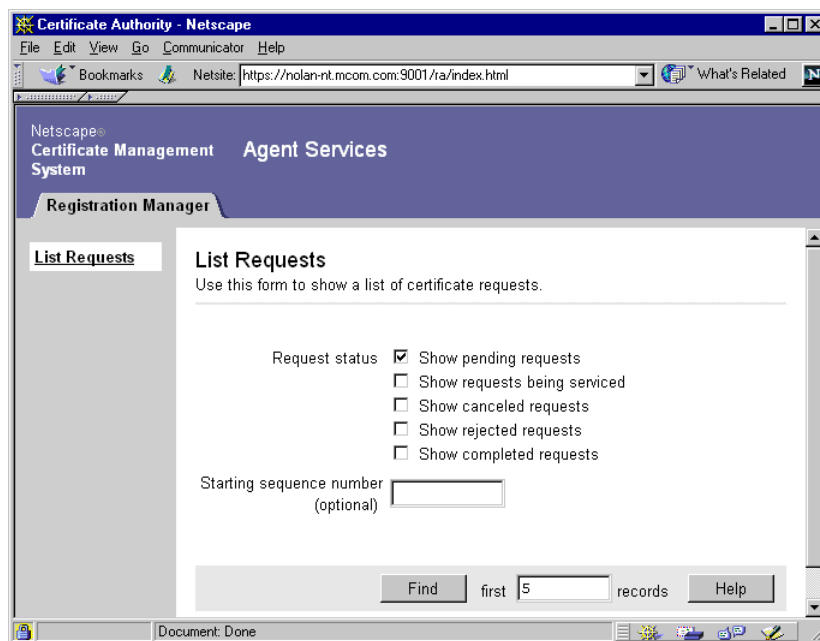
Using the default forms, a Certificate Manager agent can accomplish tasks such as these:

- Listing *deferred* certificate requests from end entities and process them
- Listing certificates issued by the server
- Searching for certificates issued by the server
- Revoking certificates issued by the server
- Updating certificates and certificate revocation lists (CRLs) maintained in the publishing directory

## Registration Manager Agent Services

The Registration Manager Agent Services interface enables a Registration Manager agent to interact with the Registration Manager (the server). Figure 20.3 shows the Registration Manager Agent Services interface.

Figure 20.3 Registration Manager Agent Services interface

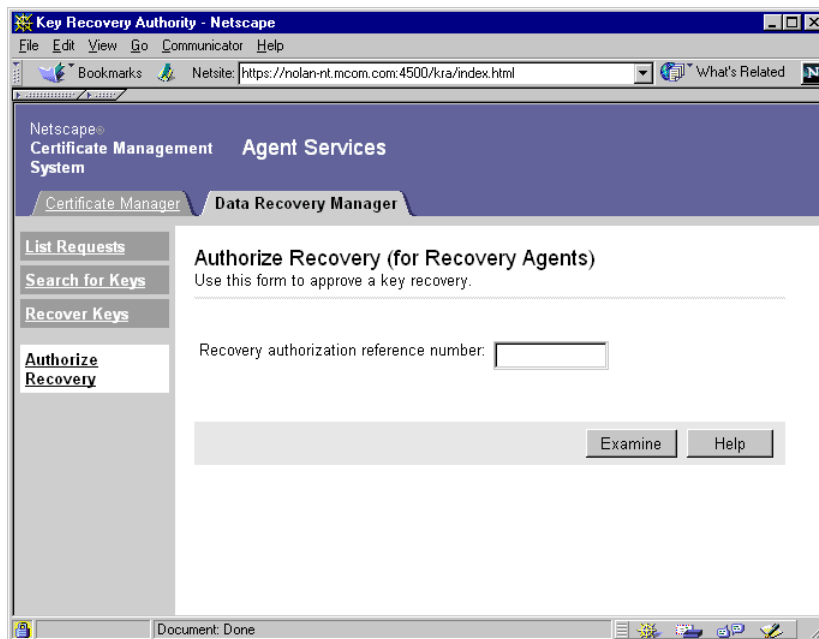


Using the default forms, a Registration Manager agent can list *deferred* certificate requests from end entities and process them.

## Data Recovery Manager Agent Services

The Data Recovery Manager Agent Services interface enables a Data Recovery Manager agent to interact with the Data Recovery Manager (the server). Figure 20.4 shows the Data Recovery Manager Agent Services interface.

Figure 20.4 Data Recovery Manager Agent Services interface



Using the default forms, a Data Recovery Manager agent can search for and recover end users' encryption private keys from the key archive. (Key recovery requires authorization from key recovery agents; see "Key Recovery Process" on page 629.)

## Accessing the Agent Services Interface

Access to the Agent Services interface is restricted to authorized agents only. For details, see "Agents" on page 135.

To access the Agent Services interface for a particular subsystem:

1. Open a web browser.
2. Go to the page where the Agent Services interface for Certificate Management System is installed.



The default URL for this page is:

`https://<host_name>:<agent_port>`

`<host_name>` is in the form `<machine_name>.<your_domain>.<domain>`

If you have customized Certificate Management System, go to the page containing the agent forms that you would use to submit a request.

3. In the Agent Services menu, choose the agent services you require:
  - To access the agent services for the Certificate Manager, click the Certificate Manager Agent Services link.
  - To access the agent services for the Registration Manager, click the Registration Manager Agent Services link.
  - To access the agent services for the Data Recovery Manager, click the Data Recovery Manager Agent Services link.

The appropriate interface appears.



# Customizing End-Entity and Agent Interfaces

The end-entity registration services interface that comes with Netscape Certificate Management System (CMS) makes it possible for end entities to interact with the server. Your end entities can use the interface's HTML-based forms to carry out various certificate and key-related operations: enroll for a certificate; and renew and revoke their certificates.

You can use the default forms as they are, customize them, or develop your own forms to suit your organization's policies or terminology. This chapter explains how the forms included in the end-entity services interface work and how to customize them.

The chapter has the following sections:

- What You Need to Know (page 548)
- How the Forms Work (page 549)
- Summary of End-Entity Forms and Templates (page 553)
- Summary of Agent Forms and Templates (page 563)

# What You Need to Know

Changing the default forms' appearance requires only a basic knowledge of HTML and a text editor. However, more significant customizing efforts require a good understanding of the following:

- HTTP, Query URLs, and HTML Forms
- JavaScript
- The way End-Entity Services interface works

## HTTP, Query URLs, and HTML Forms

Requests from the end-entity services interface to Certificate Management System are submitted using the HTTP `GET` and `POST` methods. These requests take the form of query URLs (in the case of the `GET` method) or data sent through standard output (in the case of the `POST` method).

For background on these topics, consult books on the authoring for the World Wide Web, HTML, and the HTML specification. Another good source is Netscape's documentation on creating net sites, available at this URL:

`http://home.netscape.com/assist/net\_sites/index.html`

## JavaScript

JavaScript is a simple, lightweight scripting language that most browser software, including Netscape Navigator and Communicator, use for dynamic forms. (JavaScript should not be confused with the more sophisticated and powerful Java language that is also supported by Netscape clients.)

In the end-entity services forms, JavaScript is used to check input values and to formulate requests to the server. JavaScript is also used in the output templates to present and format responses from the server.

To customize the forms and templates, you need to be familiar with JavaScript. For more information on JavaScript and its use in Netscape browsers, see the *Netscape JavaScript Authoring Guide* available at this URL:

```
http://home.netscape.com/eng/mozilla/3.0/handbook/javascript/
index.html
```

There are also several books on this topic.

## How the Forms Work

Certificate Management System provides both normal web service and certificate service operations.

Certificate service operations are requested using the HTTP and HTTPS (HTTP over SSL) protocol using the GET method with query URLs or the POST method with URL-encoded form submissions (with the usual content-type `application/x-www-form-urlencoded`). In either case, the submission results in a set of named attribute-value pairs that constitutes the request.

For certificate service operations, the URI portion should indicate

```
/<operation>
```

where `operation` designates the certificate (management) service portion, such as enrollment, retrieval, renewal, or revocation of the server. Any HTTP operations with URIs that do not begin with the `/<operation>` prefix are treated as requests for other kinds of web service by the server.

## Requests Sent to the Server

The end-entity services interface consists of a set of operations. Each operation has a name and expects a specific set of named parameters.

Each certificate service operation must contain the attribute `op` in the submission. The `op` attribute identifies the specific certificate service operation that is being requested. Each operation has required and optional parameters that must be included among the remaining attributes and values in the submission. The parameters used by a given operation are specific to that operation.

(The examples in this chapter assume that Certificate Management System is running on the host `certs.mydom.com` and is listening on the standard HTTPS port.)

The following example illustrates a sample request sent through the certificate service interface. This request executes the `displayBySerial` operation, which displays a certificate with a given serial number. The `displayBySerial` operation uses a single required input parameter, `serialNumber`. This parameter is the serial number of the certificate to be displayed.

The following URL uses the GET method to invoke the `displayBySerial` operation on the Certificate Management System. The operation displays the certificate with the hexadecimal serial number `3a` (or `58`, in decimal).

```
https://certs.mydom.com/cms?op=displayBySerial&serialNumber=0x3a
```

You could embed this URL in a link on an HTML page to allow users to view this certificate.

If you used an HTML form (rather than a query URL) to invoke this operation, the form might look like this:

```
<FORM ACTION="https://certs.mydom.com/cms" METHOD="POST">
  <INPUT TYPE="HIDDEN" NAME="op" VALUE="displayBySerial">
  Serial Number: <INPUT TYPE="TEXT" NAME="serialNumber">
  <INPUT TYPE="SUBMIT" VALUE="Display This Certificate">
</FORM>
```

In this form-based version, the user can enter the serial number of the certificate to be displayed.

## Responses and Output Templates

Certificate Management System responds to certificate service requests by sending back an HTML page built from two parts: a fragment of JavaScript code containing the data resulting from the operation and a template defining how the data is processed and displayed.

The fragment of JavaScript code consists of a `result` object that contains data properties only (no methods). The properties of the object correspond to parts of the response.

The template generally contains a combination of HTML and JavaScript code that processes and displays data. The template is set up to make use of the data in the `result` object.

In responding to a request, Certificate Management System determines the data that needs to be returned, embeds the data in the definition of the `result` object, and inserts the `result` object in the template. When the browser receives the constructed HTML page from the server, the JavaScript code in the template file looks at the values in the `result` object and uses the data to display the HTML page to the user.

Because the functions that display and manipulate the data are accessible to you in the plain-text template files (as opposed to being hardcoded in the server's libraries), you can customize the way data is used and presented to the user by editing the JavaScript and HTML in the template files.

For example, the `displayBySerial` operation generates the following JavaScript code fragment:

```
<SCRIPT LANGUAGE="JavaScript">

var header = new Object();

var result = new Object();

header.certPrettyPrint = [long string containing pretty-
printed certificate]

header.certChainBase64 = [string containing base64-encoded
certificate]

header.serviceURL = "https://tahoma.mcom.com/cms";

header.serialNumber = 2;

result.header = header;

</SCRIPT>
```

Notice how this code fragment defines an object named `result` and puts the resulting data from the operation in the properties of that object. Each certificate service operation returns an object named `result`. The contents of the `result` object are specific to the operation.

When it responds to the request, Certificate Management System inserts this JavaScript fragment into the template file for the `displayBySerial` operation. The server inserts the fragment in the position in the template file where it finds the server-parsed tag `<CMS_TEMPLATE>`. It then returns the completed template with the inserted fragment to the client. The client then processes the completed template and displays the resulting page. In the case of the `displayBySerial` operation, the template file uses JavaScript and HTML to display the contents of the `result` object to the user.

Because the data from the operation is available in the `result` object, you can customize the JavaScript in the template or write your own functions to use this data. For example, to access the certificate's serial number, you can write a JavaScript function that uses `result.header.serialNumber`.

Templates for each operation are stored in the `forms` subdirectory of the server instance. Each operation has a corresponding template file in this directory. The template filename is as follows:

`<operation>.template`

`<operation>` is the name of the operation. For example, the template file for the `displayBySerial` operation is `displayBySerial.template`.

**Important** Be careful not to confuse the `forms` directory for the server instance (`<server_root>/cert-<instance_id>/web`) with the distribution-source forms directory (`<server_root>/bin/cert/forms`). You should edit the forms for a server instance only. Do not edit the source forms in the `<server_root>/bin/cms/forms` directory.

The server reads the templates dynamically; you do not have to restart the server for it to read any changes to the template files.



## Errors and the Error Template

All certificate service errors in the end-entity interface are returned through a single template called `GenError.template`. The `error result` object contains the following data properties:

```
<SCRIPT LANGUAGE="JavaScript">

var header = new Object();

var result = new Object();

header.errorDetails = [ a string describing the context of
the error ]

header.errorDescription = [ a string describing the error ]

result.header = header;

</SCRIPT>
```

The error template included in the distribution prints the information in the `error result` object along with some explanatory text.

## Summary of End-Entity Forms and Templates

This section gives the location of the default forms that appear in the end-entity services interface and lists the forms by type of end-entity operation. Figure 21.1 shows the end-entity interface of a Registration Manager.

The end-entity services interface is divided into three parts or frames—top, menu, and content. The top frame includes tabs that are specific to end-entity operations, such as certificate enrollments and renewals. The menu lists all the

operations supported by the selected tab. The content shows the form pertaining to the operation an end entity chooses in the menu; the form contains information to carry out the selected operation.

Figure 21.1 End-entity services interface

**Certificate Management System - Netscape**

File Edit View Go Communicator Help

Bookmarks Netsite: <http://supriya-nt.mcom.com:90/> What's Related

---

**Registration Manager**

**Enrollment** Renewal Revocation Retrieval

---

**User Enrollment**

Manual

Directory Based

**Directory and PIN Based**

---

**Server Enrollment**

Manual

Directory Based

---

**Registration Manager Enrollment**

Manual

---

**Certificate Manager Enrollment**

---

**Directory And PIN Based User Enrollment**

Use this form to submit a request for a personal certificate based on information supplied by your organization's directory. Your user ID and password for the directory and a one time personal identification number (PIN) assigned by your system administrator are required for this automatic method of certificate issuance. If the user ID, password and PIN are correct your certificate will be issued automatically.

---

**Important:** Be sure to request your certificate on the same computer on which you plan to use the certificate.

---

**User's Identity**

Enter your user ID and password for your organization's directory and the one time PIN given by your system administrator. This information will be used to verify your identity and to obtain information from the directory to fill in the certificate.

User Id:

Password:

Enter the PIN your system administrator has communicated to you for certificate enrollment.

PIN:

---

**Key Length Information**

When you submit this form, the browser generates a private key and a public key. It retains the private key and submits the public key along with your request for a

Document: Done

# Locating End-Entity Forms and Templates

You can find the HTML forms specific to end-entity operations and the corresponding output templates at this location:

```
<server_root>/cert-<instance_id>/web/ee/...
```

<server\_root> is the directory where the CMS binaries are kept. You first specified this directory during installation.

<instance\_id> is the ID for this instance of Certificate Management System. You first specified this during installation.

## Forms for Certificate Enrollment

Table 21.1 lists the forms that correspond to the menu options in the Enrollment tab of the end-entity services interface.

Table 21.1 Default forms for end-entity enrollment

Menu link and form filename	Description
<b>User Enrollment</b> (lists menu options for end-user enrollment)	
Manual (ManUserEnroll.html)	End users can use this form to request SSL client and S/MIME certificates. Both the Certificate Manager and Registration Manager provide this form.
Directory Based (DirUserEnroll.html)	End users can use this form to request SSL client and S/MIME certificates. Both the Certificate Manager and Registration Manager provide this form.
Directory and PIN Based (DirPinUserEnroll.html)	End users can use this form to request SSL client and S/MIME certificates. Both the Certificate Manager and Registration Manager provide this form.
<b>Server Enrollment</b> (lists menu options for server enrollment)	

Table 21.1 Default forms for end-entity enrollment (Continued)

Menu link and form filename	Description
Manual (ManServerEnroll.html)	Server administrators can use this form to request SSL server certificates for servers. Both the Certificate Manager and Registration Manager provide this form.
Directory Based (DirServerEnroll.html)	Server administrators can use this form to request SSL server certificates for servers. Both the Certificate Manager and Registration Manager provide this form.
	This form is not used in the default interface.
<b>Registration Manager Enrollment</b> (lists menu options for Registration Manager enrollment)	
Manual (ManRAEnroll.html)	Registration Manager administrators can use this form to request a <i>signing certificate</i> for a Registration Manager. For details about this certificate, see “Signing Key Pair and Certificate” on page 188.
	Both the Certificate Manager and Registration Manager provide this form.
<b>Certificate Manager Enrollment</b> (lists menu options for Certificate Manager enrollment)	
Manual (ManCAEnroll.html)	Certificate Manager administrators can use this form to request <i>CA signing certificates</i> for Certificate Managers functioning as subordinate CAs. For details about the CA signing certificate, see “CA Signing Key Pair and Certificate” on page 184.
	Only the Certificate Manager provides this form.
<b>Object Signing Enrollment</b> (lists menu options for object signing enrollment)	
Manual (ManObjSignEnroll.html)	End users and administrators can use this form to enroll for a certificate that allows them to sign objects, such as Java applets. Both the Certificate Manager and Registration Manager provide this form.

## Forms for Certificate Renewal

Table 21.2 lists the forms that correspond to the menu options in the Renewal tab of the end-entity services interface.

**Table 21.2** Default forms for certificate renewal

Menu link and form filename	Description
Server Certificate (ServerRenewal.html)	Server administrators can use this form to renew server certificates. Both the Certificate Manager and Registration Manager provide this form.  This form is not used in the default interface.
User Certificate (UserRenewal.html)	End users can use this form to renew their SSL client and S/MIME certificates; users can renew their S/MIME certificates, only if these certificates were issued with the SSL client bit set. Both the Certificate Manager and Registration Manager provide this form.

## Forms for Certificate Revocation

Table 21.3 lists the forms that correspond to the menu options in the Revocation tab of the end-entity services interface.

**Table 21.3** Default forms for certificate revocation

Menu link and form filename	Description
Server Revocation (ServerRevocation.html)	Server administrators can use this form to revoke server certificates. Both the Certificate Manager and Registration Manager provide this form.  This form is not used in the default interface.
User Revocation (UserRevocation.html)	End users can use this form to revoke their SSL client certificates. Both the Certificate Manager and Registration Manager provide this form.

# Forms for Certificate Retrieval

Table 21.4 lists the forms that correspond to the menu options in the Retrieval tab of the end-entity services interface.

Table 21.4 Default forms provided for certificate retrieval

Menu link and form filename	Description
List Certificates (queryBySerial.html)	End users and administrators can use this form to list certificates based on their serial numbers. Only the Certificate Manager provides this form.
Search for Certificates (queryCert.html)	<div>End users and administrators can use this form to search for specific certificates. The search criteria can be a combination of the following:</div> <ul style="list-style-type: none"><li>• Serial number of the certificate</li><li>• Subject name of the certificate</li><li>• Revocation status of the certificate</li><li>• Issuing Information—when the certificate was issued</li><li>• Validity period of the certificate</li><li>• Type of certificate</li></ul> <div>Only the Certificate Manager provides this form.</div>

Table 21.4 Default forms provided for certificate retrieval (Continued)

Menu link and form filename	Description
Import CA Certificate Chain (GetCACChain.html)	<p>End users and administrators can use this form to import the certificate chain of a Certificate Manager (CA) into their browsers or servers. They can</p> <ul style="list-style-type: none"><li>• Import the CA certificate chain into their browsers</li><li>• Download the CA certificate chain in binary form</li><li>• View the CA certificate chain for importing into a server</li><li>• Display certificates in the CA certificate chain for importing individually into a server</li></ul> <p>Both the Certificate Manager and Registration Manager provide this form.</p>
Import Certificate Revocation List (DisplayCRL.html)	<p>End users and administrators can use this form to:</p> <ul style="list-style-type: none"><li>• Manually check the revocation status of a particular certificate (if they are not sure whether they have the latest version of the CRL)</li><li>• Import the latest CRL to your Netscape Navigator</li><li>• Download the latest CRL in binary form</li><li>• View the CRL header information</li></ul> <p>Only the Certificate Manager provides this form.</p>

## Forms for Key Recovery

Table 21.5 lists the forms that correspond to the menu options in the Recovery tab of the end-entity services interface.

Table 21.5 Default forms for encryption private key recovery

Menu link and form filename	Description
Key Recovery (KeyRecovery.html)	End users can use this form to retrieve their encryption private keys from the Data Recovery Manager. Both the Certificate Manager and Registration Manager provide this form, if you configure them to function as trusted managers to a Data Recovery Manager.  Note that end-user initiated key recovery is not supported in this version of the software. The form is not used in the default interface.

## Other Forms

Table 21.5 lists common forms that are used by the operation-specific forms in the end-entity services interface.

Table 21.6 Default forms for encryption private key recovery

Form filename	Description
enrollMenu.html	This file loads and highlights the Enrollment tab.
renewalMenu.html	This file loads and highlights the Renewal tab.
recoveryMenu.html	This file loads and highlights the Recovery tab.
retrievalMenu.html	This file loads and highlights the Retrieval tab.
index.html	This file contains the menu options. To change the name of an option, search for it in the file and then edit it.



Table 21.6 Default forms for encryption private key recovery (Continued)

Form filename	Description
*.js	Files with a .js file extension include JavaScript helper functions that are used by all the other forms provided for end entities.
xenroll.dll	This file enables the end user enrollment forms to work with Microsoft Internet Explorer.

## Output Templates for End-Entity Operations

Table 21.7 lists the default templates that are displayed when a user performs a certificate operation. For example, when searching for certificates.

Table 21.7 Summary of default templates for the end-entity services interface

Template filename	Description
displayBySerial.template	Used to display information pertaining to a certificate when users view an individual certificate (for example, when they click the Details button next to a certificate).
EnrollSuccess.template	Used to inform the CMS administrator that the agent certificate he or she requested has been successfully installed in the subsystem's internal database.
GenError.template	Used to display an error message to the user when an error occurs.
GenPending.template	Used to inform the user that the certificate request he or she submitted has been queued for agent approval.

Table 21.7 Summary of default templates for the end-entity services interface (Continued)

Template filename	Description
<code>GenRejected.template</code>	Used to inform the user that the certificate request he or she submitted has been rejected by the server.
<code>GenSuccess.template</code>	Used to inform the user that the certificate request he or she submitted has been approved by the server.
<code>GenSvcPending.template</code>	Used to inform the user that the certificate request he or she submitted has been queued for agent approval.
<code>GenUnauthorized.template</code>	Used to inform the user that he or she performed an authorized operation.
<code>GenUnexpectedError.template</code>	Used to inform the user that the server encountered an unexpected error while processing the request she or he submitted.
<code>ImportCert.template</code>	Used to display the CA certificate when users attempt to import the CA certificate.
<code>queryCert.template</code>	Used to display the list of certificates when users search for certificates.
<code>RenewalSuccess</code>	Used to inform the user that the certificate she or he submitted for renewal has been successfully renewed.
<code>RevocationSuccess</code>	Used to inform the user that the certificate she or he submitted for revocation has been revoked.

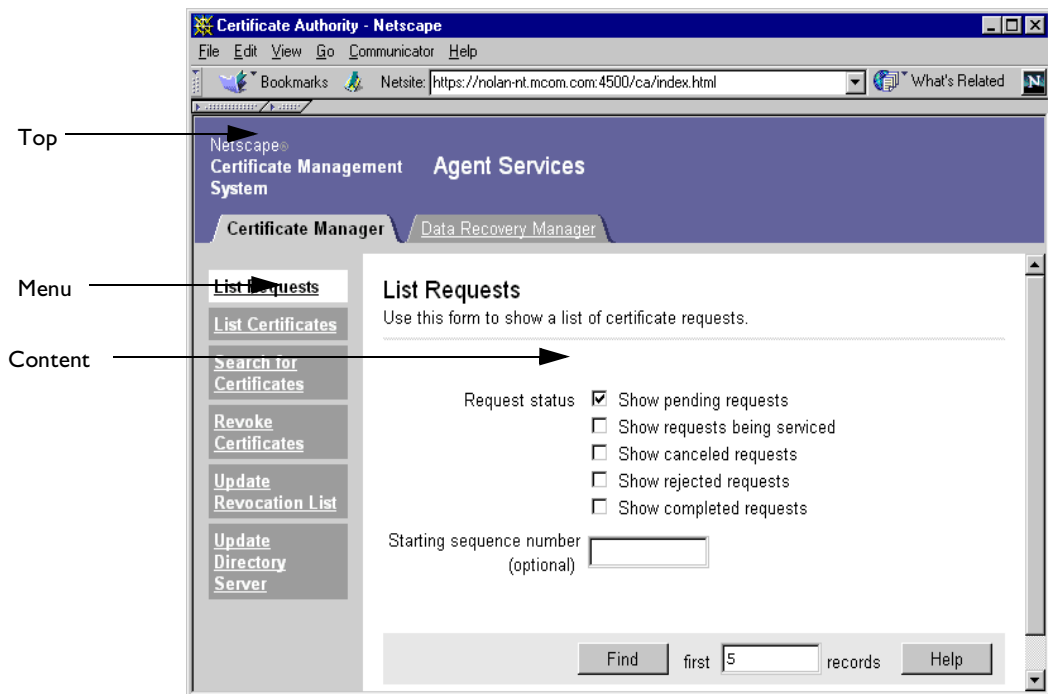
# Summary of Agent Forms and Templates

This section describes the Agent Services interface, gives the location of the agent forms and output templates, and lists all of the default forms and templates.

## Structure of the Agent Services Interface

As shown in Figure 21.2, the Agent Services interface is divided into three parts or frames—top, menu, and content. The top frame includes the tabs that allow you to select a subsystem. The menu lists all the operations supported by the selected subsystem. The content shows the form pertaining to the operation an agent chooses in the menu; the form contains information to carry out the selected operation.

Figure 21.2 Various parts of the Agent Services interface



## Locating Agent Forms and Templates

You can find the HTML forms specific to agent operations and the corresponding output templates at this location:

```
<server_root>/cert-<instance_id>/web/agent/<subsystem>/...
```

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

`<instance_id>` is the ID for this instance of Certificate Management System. You first specified this during installation.

`<subsystem>` refers to the forms directory pertaining to a subsystem, the Certificate Manager (ca), Registration Manager (ra), or Data Recovery Manager (kra).

# *Logs*



*Chapter 22* Introduction to Logs

*Chapter 23* Managing Logs



# Introduction to Logs

Netscape Certificate Management System (CMS) creates log files that record events related to its activities, such as administration, communications using any of the protocols the server supports, and various other processes employed by all the subsystems the server manages.

This chapter identifies various logs maintained by Certificate Management System and describes them in detail. The chapter has the following sections:

- Logs Maintained by Certificate Management System (page 568)
- Services That Are Logged (page 569)
- Log Levels (Message Categories) (page 570)
- Log File Locations (page 572)
- Log File Naming Conventions (page 573)
- Buffered Versus Unbuffered Logging (page 574)
- Rotation of Log Files (page 574)
- Deletion of Log Files (page 575)
- Archiving of Rotated Log Files (page 576)

# Logs Maintained by Certificate Management System

While Certificate Management System is running, it keeps a log of information and error messages on all the components it manages. These messages are broadly categorized into three separate logs and are maintained in three separate log files, as listed in Table 22.1.

During installation, Certificate Management System automatically creates the required log files in your local file system. The server creates common system, error, and audit files for all components that were installed together, and it logs messages to these files. For example, if you installed a Certificate Manager and a Data Recovery Manager together, you will find log messages for both the subsystems in the same log file.

Table 22.1 Types of logs maintained by Certificate Management System

Log type	Description
System	<p>This log records information about requests to the server (all HTTP and HTTPS requests) and the responses from the server. Information recorded in this log includes the IP address of the client machine that accessed the server, operations performed (for example, search, add, edit), and the result of the access (for example, the number of entries returned). This log is on by default.</p> <p>For more information, see “Monitoring System Logs” on page 587.</p>
Error	<p>This log contains the error messages the server has encountered (HTTP errors and errors with the certificate service). This log is on by default.</p> <p>For more information, see “Monitoring Error Logs” on page 589.</p>



Table 22.1 Types of logs maintained by Certificate Management System (Continued)

Log type	Description
Audit	This log records messages specific to the certificate service—messages such as certificate requests, certificate renewal and revocation requests, and CRL publication—and enables you to detect any unauthorized access or activity. This log is on by default.  For more information, see “Monitoring Audit Logs” on page 592.

## Services That Are Logged

All major components and protocols (or services) of Certificate Management System log messages to log files. Table 22.2 lists services that are logged by default. If you want to view messages logged by a specific service, you can customize log settings accordingly. For details, see “Monitoring Logs” on page 586.

Table 22.2 Services logged by Certificate Management System

Service	Description
All	Specifies logged events related to all the services.
Registration Authority	Specifies logged events related to the Registration Manager.
Certificate Authority	Specifies logged events related to the Certificate Manager.
Key Recovery Authority	Specifies logged events related to the Data Recovery Manager.
HTTP	Specifies logged events related to the HTTP activity of the server.

Table 22.2 Services logged by Certificate Management System (Continued)

Service	Description
Database	Specifies logged events related to this server's activity with the internal database.
Authentication	Specifies logged events related to this server's activity with the authentication module.
Administration	Specifies logged events related to this server's administration activities—that is, HTTPS communication between the CMS window and Certificate Management System.
LDAP	Specifies logged events related to this server's activity with the LDAP directory (used for publishing certificates and CRLs).
Request Queue	Specifies logged events related to the request queue activity of this server.
ACLs	Specifies logged events related to access control lists.
User and Group	Specifies logged events related to users and groups managed by this server.
Others	Specifies logged events related to other activities of this server, such as command-line utilities and other processes.

## Log Levels (Message Categories)

For identification and filtering purposes, events logged by all CMS-supported services are classified into various categories. These are listed in Table 22.3. Each category represents messages that are of the same or a similar nature or that belong to a specific functional area. A particular log, for example the error log, can record entries that fall under one or more of these categories.

In the CMS configuration, each message category corresponds to a specific log level. Log levels are represented by numbers (digits) 1 to 6, each digit indicating the level of logging to be performed by the server—that is, how detailed the logging should be.

- A higher priority level (a larger digit) means less detail because only events of high priority are logged.
- A lower priority level (a smaller digit) means greater detail because more kinds of events are recorded in the log file.

**Table 22.3** Classification of log entries or messages

Log level	Message category	Description
0	Debugging	These messages contain debugging information.
1	Informational	These messages provide general information about the state of Certificate Management System. For example, status messages such as “Certificate Management System initialization complete” and “Request for operation succeeded” fall into this category.
2	Warning	These messages are warnings only and do not indicate any failure in the normal operation of the server.
3	Failure (default)	<p>These messages indicate errors and failures that prevent the server from operating normally.</p> <p>Examples of messages that fall into this category include failures to perform a certificate service operation (“User authentication failed” or “Certificate revoked”) and unexpected situations that can cause irrevocable errors (“The server cannot send back the request it processed for a client through the same channel the request came from the client”).</p>
4	Misconfiguration	These messages indicate that a misconfiguration in the server is causing an error.

Table 22.3 Classification of log entries or messages (Continued)

Log level	Message category	Description
5	Catastrophic failure	These messages indicate that because of an error, the service cannot continue running.
6	Security-related events	These messages identify occurrences that affect the security of the server (for example, "Privileged access attempted by user with revoked or unlisted certificate").

You can use log levels to filter log entries based on the severity of an event. By default, a level 3 (Failure) is set for all services.

**Important** The log level is additive—that is, specifying a value of 3 causes levels 4, 5, and 6 to be logged. Log data can be voluminous, especially at lower (more verbose) logging levels. Make sure that the host machine has sufficient disk space for all the log files. It is also important to define your logging level, log rotation, log expiration, and server-backup policies appropriately so that all the log files are backed up and the host system doesn't get overloaded; otherwise, you may lose information.

## Log File Locations

For quick access, all the log files—system, error, and audit—are maintained in your local file system. Make sure that your storage capacity is sufficient for all your log files. A log file has the following default location:

`<server_root>/cert-<instance_id>/logs/...`

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

`<instance_id>` is the ID for this instance of Certificate Management System. You first specified this during installation.

You can change the default location for logs by modifying it in the configuration. For details, see “Log Parameters in the Configuration File” on page 580.

# Log File Naming Conventions

All log files created by Certificate Management System use one or the other of two naming conventions. There is one naming convention for active log files and one for rotated log files.

## Active Log File Naming Convention

All active log files created by Certificate Management System use an identical naming convention. The name of an active log file is in the form `<log_type>.log`, where `<log_type>` specifies the log file type—whether it is system, error, or audit.

For example, an active error log file would be named `error.log`.

## Rotated Log File Naming Convention

All rotated log files created by Certificate Management System use an identical naming convention. When Certificate Management System rotates an active log file, it renames the current log file and then creates a new log file with the original name. The rotated log file is saved with the original file type and an appended timestamp.

The name of a rotated log file is in the form `<log_type>.timestamp`, where the components of the filename indicate the following:

- `<log_type>` specifies the log file type—system, error, or audit—that has been rotated.
- `timestamp` is a large integer that indicates the date and time the corresponding active log file was rotated. The date and time have the forms `YYYYMMDD` (Year, Month, Day) and `HHmmSS` (Hour, Minute, Second), in that order.

For example, an error log file rotated on July 28, 1998 at 12:36:24 would be named `error.19980728123624`.

**Note** The timestamp is expressed in standard Unix time: the number of seconds since midnight January 1, 1970.

## Buffered Versus Unbuffered Logging

Certificate Management System supports buffered logging for all three types of logs—system, error, and audit. You can choose to configure the server for either buffered or unbuffered logging (see “Configuring Logs” on page 581).

If you configure Certificate Management System for buffered logging, the server creates buffers for the corresponding logs, and it holds the messages in these buffers for as long as possible. The server flushes out the messages to the log files—which are maintained in your local file system—only when either of the following conditions occurs:

- The buffer gets full—the buffer gets full when the buffer size is equal to or greater than the value specified by the `bufferSize` configuration parameter. The default value for this parameter is 512 KB.
- The flush interval for the buffer is reached—the flush interval is reached when the time interval since the last buffer flush is equal to or greater than the value specified by the `flushInterval` configuration parameter. The default value for this parameter is 300 seconds.
- When current logs are read from CMS window—the server retrieves the latest log when it is queried for current logs.

If you configure the server for unbuffered logging, the server flushes out messages as they are generated to the log files. Because the server performs an I/O operation (writing to the log file) each time a message is generated, configuring the server for unbuffered logging decreases performance.

## Rotation of Log Files

Certificate Management System supports automatic rotation of log files, which simplifies administration and facilitates backups. You are not required to manually retire the current log file and create a new one to hold subsequent logged events. You can back up all but the current log file in a directory at any time, without stopping the server or manually notifying the server to start a new log file. The parameters that control log rotation are specified in the configuration. To change the log file rotation parameters, see “Configuring Logs” on page 581.

You should periodically archive or back up the rotated log files. For details, see “Archiving of Rotated Log Files” on page 576.

## Timing of Log File Rotation

Log files are rotated when either of the following conditions occur:

- The size limit for the corresponding file is reached—the size of the corresponding log file is equal to or greater than the value specified by the `maxFileSize` configuration parameter. The default value for this parameter is 100 KB.
- The age limit for the corresponding file is reached—the corresponding log file is equal to or older than the interval specified by the `rolloverInterval` configuration parameter. The default value for this parameter is 2592000 seconds (every hour).

## Location of Rotated Log Files

Rotated log files are stored at the same location where the current or active log files are maintained. To find out where the active log files are located, see “Log File Locations” on page 572.

## Deletion of Log Files

Certificate Management System supports automatic deletion of rotated (or old) log files. The parameters that control log deletion are specified in the configuration file.

## How to Conserve Disk Space

By default, Certificate Management System does not delete rotated log files automatically. Because the rotated log files are also saved in your local file system, these files eventually take up a considerable amount of disk space. You can avoid this problem by doing one of the following:

- Configure the server to automatically delete the rotated log files.
- Manually delete the log files from the local file system.

In either case, if you want to keep specific log files for future use, be sure to archive or back them up before they are deleted. For details, see “Archiving of Rotated Log Files” on page 576.

## Timing of Log File Deletion

If you configure Certificate Management System to delete rotated log files automatically, the server deletes these files when the life of the corresponding log file is equal to or older than the interval specified by the `expirationTime` configuration parameter. The default value for this parameter is 2592000 seconds (or every hour); see “Log Parameters in the Configuration File” on page 580.

## Archiving of Rotated Log Files

Log files, especially the audit log file, contain critical information. So it is good practice to periodically archive rotated log files to some archive media. Consider doing this whether you are manually deleting rotated log files or have configured the server to delete files automatically. You can archive log files by copying the entire `log` directory to your archive media.

Certificate Management System does not provide any tool or utility for archiving log files. Use the tools or utilities that your operating system provides for archiving.

Certificate Management System does, however, provide a command-line utility, called `signtool`, that allows you to sign log files before archiving them. This gives you a means of tamper detection. For details, see “Signing Log Files” on page 597.



## Managing Logs

Each instance of Netscape Certificate Management System (CMS) maintains its own system, error, and audit log files. These files record events related to various CMS activities. By configuring logs, you can customize the contents in the log files.

This chapter explains how to use the CMS window to configure the system, error, and audit logs maintained by Certificate Management System, and how to monitor its activities by viewing log contents.

Before you attempt to configure or monitor logs, it's a good idea to read "Introduction to Logs" on page 567.

The chapter has the following sections:

- Management of Logs (page 578)
- Configuring Logs (page 581)
- Monitoring Logs (page 586)
- Signing Log Files (page 597)

# Management of Logs

You can manage CMS logs in two ways:

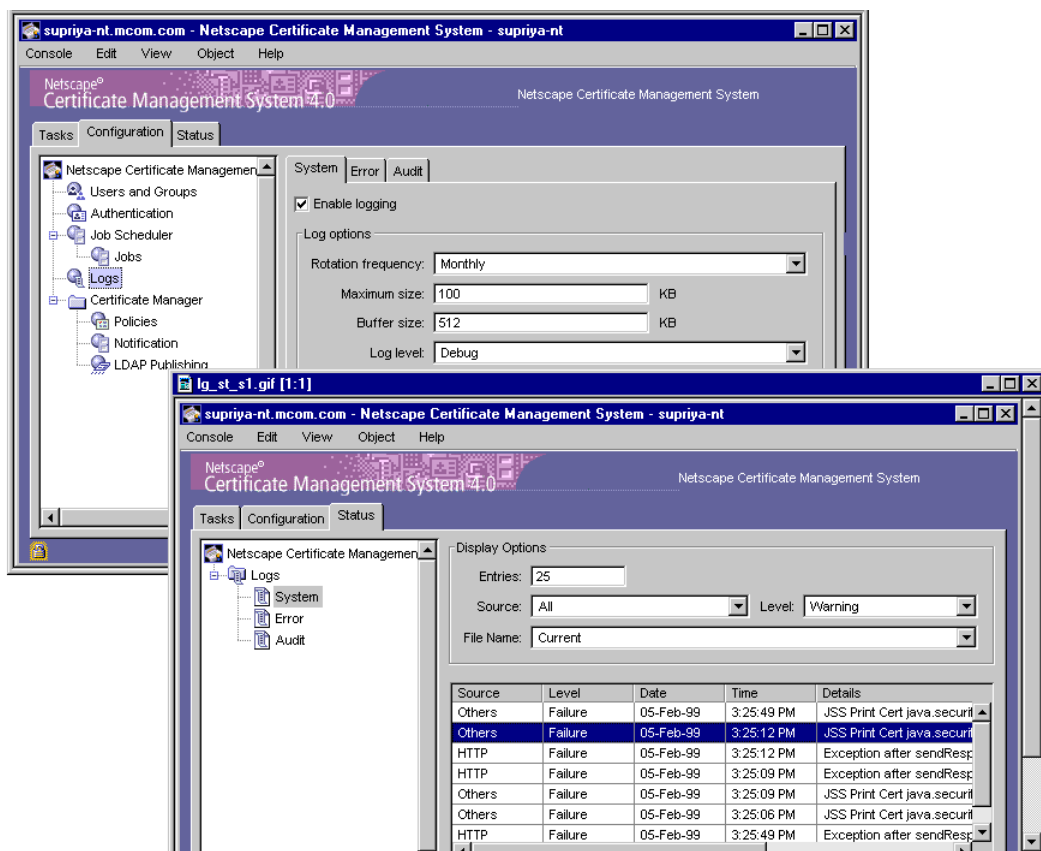
- By making changes to logging-specific configuration parameters from the CMS window and monitoring the resulting messages in the CMS window. See “Log Management from the CMS Window” on page 578.
- By editing the configuration parameters in the configuration file. See “Log Parameters in the Configuration File” on page 580.

The recommended method is to use the CMS window. However, for configuration parameters that are not shown in the CMS window, you may have to edit the configuration file.

## Log Management from the CMS Window

The CMS window supports the configuration and monitoring of various CMS logs. In this window, you will find the Logs object in two places—in the navigation tree of the Configuration tab and in the navigation tree of the Status tab (see Figure 23.1).

Figure 23.1 Managing logs from the CMS window



The Logs object in the Configuration tab shows the current configuration of system, error, and audit logs and allows you to change it. For instructions on changing log configurations, see “Configuring Logs” on page 581.

The Logs object in the Status tab shows messages logged by the server. For instructions on viewing logs, see “Monitoring Logs” on page 586.

## Log Parameters in the Configuration File

The sample in Figure 23.2 illustrates how information specific to logs appears in the configuration file, `CMS.cfg`. If you intend to change the configuration by editing the configuration file, be sure to follow the instructions provided in “Changing the Configuration by Editing the Configuration File” on page 72.

**Figure 23.2** Information specific to logs in the CMS configuration

```
logAudit._000=##
logAudit._001=## Logging
logAudit._002=##
logAudit.bufferSize=512
logAudit.expirationTime=2592000
logAudit.fileName=C:/Netscape/Server4/cert-testCA/logs/audit
logAudit.flushInterval=300
logAudit.level=3
logAudit.maxFileSize=100
logAudit.on=true
logAudit.rolloverInterval=2592000

logError._000=##
logError._001=## Logging
logError._002=##
logError.bufferSize=512
logError.expirationTime=2592000
logError.fileName=C:/Netscape/Server4/cert-testCA/logs/error
logError.flushInterval=300
logError.level=3
logError.maxFileSize=100
logError.on=true
logError.rolloverInterval=2592000

logNTAudit.NTEventSourceName=cert-testCA
logNTAudit.level=1
logNTAudit.on=true
logNTSystem.NTEventSourceName=cert-testCA
logNTSystem.level=2
logNTSystem.on=true

logSystem._000=##
logSystem._001=## Logging
logSystem._002=##
logSystem.bufferSize=512
logSystem.expirationTime=2592000
logSystem.fileName=C:/Netscape/Server4/cert-testCA/logs/system
logSystem.flushInterval=300
logSystem.level=3
logSystem.maxFileSize=100
logSystem.on=true
logSystem.rolloverInterval=2592000
```

# Configuring Logs

This section describes the procedures for configuring each type of CMS log:

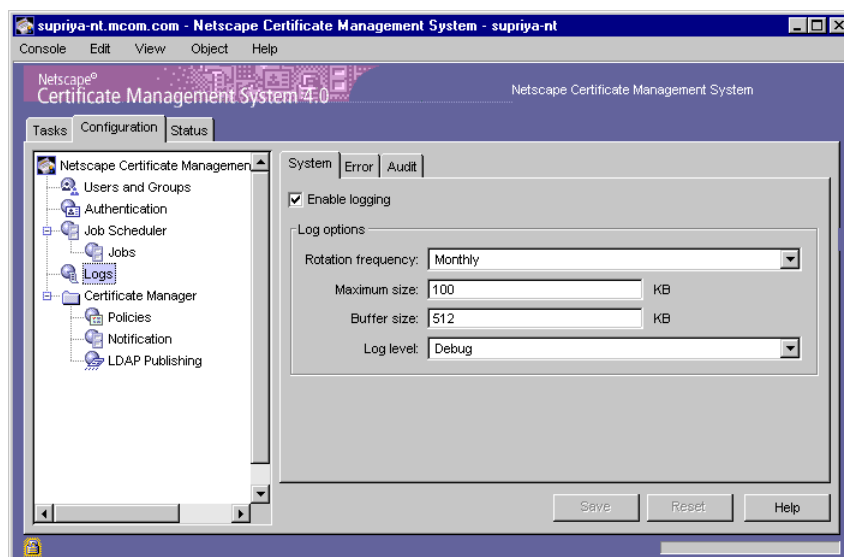
- Configuring System Logs
- Configuring Error Logs
- Configuring Audit Logs

## Configuring System Logs

To configure the system log for a CMS instance:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. In the navigation tree, click Logs.

The System tab appears in the right pane. It shows the current configuration for the system log.



3. If you want the server to log system-level messages, check the “Enable logging” box. All the associated fields become available for you to enter information. Leave the box unchecked if you do not want the server to log messages of this type.

4. In the “Log options” section, specify information as appropriate:

**Rotation frequency.** From the drop-down list, select the interval at which the server should rotate the active system log file. The available choices are Hourly, Daily, Weekly, Monthly, and Yearly. The default rotation interval is Monthly. For more information, see “Rotation of Log Files” on page 574.

**Maximum size.** Type the file size in kilobytes (KB) for the system log. The default file size is 100 KB. For more information, see “Rotation of Log Files” on page 574.

**Buffer size.** Type the buffer size in kilobytes (KB) for the system log. The default size for the buffer is 512 KB. For more information, see “Buffered Versus Unbuffered Logging” on page 574.

**Log level.** From the drop-down list, select a log level. The choices are Debug, Info, Warning, Failure, Misconfiguration, Catastrophe, and Security. For more information, see “Log Levels (Message Categories)” on page 570.

5. To save your changes, click Save.

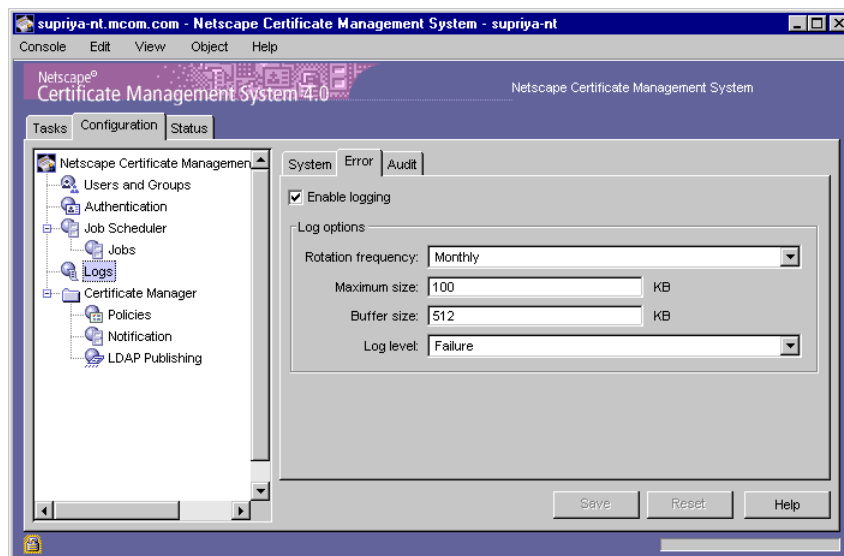
The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

## Configuring Error Logs

To configure the error log for a CMS instance:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. In the navigation tree, click Logs, and then in the right pane, click Error.

The Error tab appears, showing the current configuration of the error log.



3. If you want the server to log error messages, check the “Enable logging” box. All the associated fields become available for you to enter information. Leave the box unchecked if you do not want the server to log messages of this type.
4. In the “Log options” section, specify information as appropriate:

**Rotation frequency.** From the drop-down list, select the interval at which the server should rotate the active error log file. The available choices are Hourly, Daily, Weekly, Monthly, and Yearly. The default selection is Monthly. For more information, see “Rotation of Log Files” on page 574.

**Maximum size.** Type the file size in kilobytes (KB) for the error log. The default file size is 100 KB. For more information, see “Rotation of Log Files” on page 574.

**Buffer size.** Type the buffer size in kilobytes (KB) for the error log. The default size for the buffer is 512 KB. For more information, see “Buffered Versus Unbuffered Logging” on page 574.

**Log level.** From the drop-down list, select a log level. The choices are Debug, Info, Warning, Failure, Misconfiguration, Catastrophe, and Security. For more information, see “Log Levels (Message Categories)” on page 570.

5. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

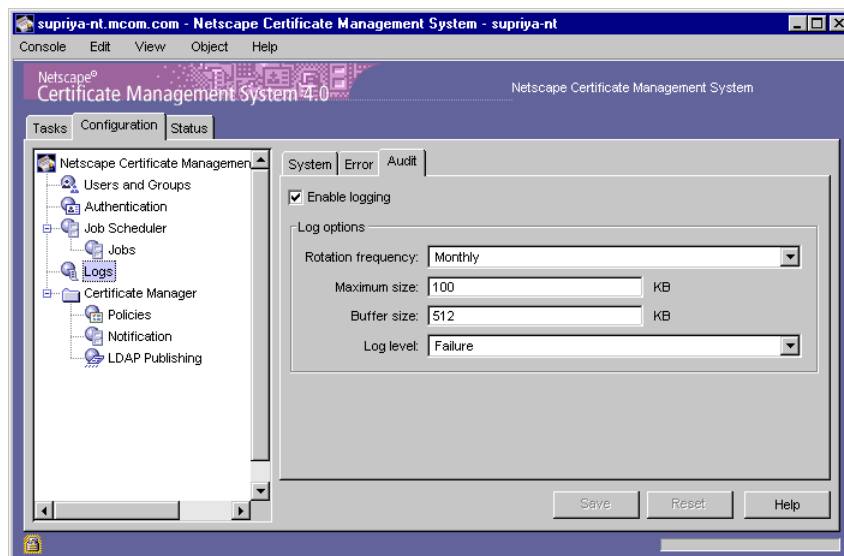


## Configuring Audit Logs

To configure the audit log for a CMS instance:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. In the navigation tree, click Logs, and then in the right pane, click Audit.

The Audit tab appears, showing the current configuration for the audit log.



3. If you want the server to log system-level messages, check the “Enable logging” box. All the associated fields become available for you to enter information. Leave the box unchecked if you do not want the server to log messages of this type.
4. In the “Log options” section, specify information as appropriate:

**Rotation frequency.** From the drop-down list, select the interval at which the server should rotate the active audit log file. The available choices are Hourly, Daily, Weekly, Monthly, and Yearly. The default selection is Monthly. For more information, see “Rotation of Log Files” on page 574.

**Maximum size.** Type the file size in kilobytes (KB) for the audit log. The default file size is 100 KB. For more information, see “Rotation of Log Files” on page 574.

**Buffer size.** Type the buffer size in kilobytes (KB) for the audit log. The default size for the buffer is 512 KB. For more information, see “Buffered Versus Unbuffered Logging” on page 574.

**Log level.** From the drop-down list, select a log level. The choices are Debug, Info, Warning, Failure, Misconfiguration, Catastrophe, and Security. For more information, see “Log Levels (Message Categories)” on page 570.

5. If you need to make any other configuration changes, go to the appropriate options and modify them.
6. To save your changes, click Save.

The CMS configuration is modified. If the changes you made require you to restart the server, you will be prompted accordingly. In that case, restart the server.

## Monitoring Logs

When you have problems with Certificate Management System that require troubleshooting, you may find it helpful to check the error or informational messages that the server has logged. Also, by examining the log files you can monitor many aspects of the server's operation.

To facilitate this, the CMS window provides a simple mechanism for viewing the contents of both currently active and rotated audit, system, and error log files. The contents of the log file you choose to view are displayed in the form of a table. Each row is allocated to a specific log entry, with columns containing information such as the date and time the message was logged, the severity of the message, and a general description of the log. Once you open a log file for viewing, you can also do the following tasks:

- Read log file contents partially (by specifying the number of entries to be displayed)
- Filter log entries for specific services (by specifying the source)

This section covers the following topics on monitoring Certificate Management System by viewing log contents:

- Monitoring System Logs
- Monitoring Error Logs
- Monitoring Audit Logs
- Using System Tools for Monitoring the Server (Windows NT Only)

## Monitoring System Logs

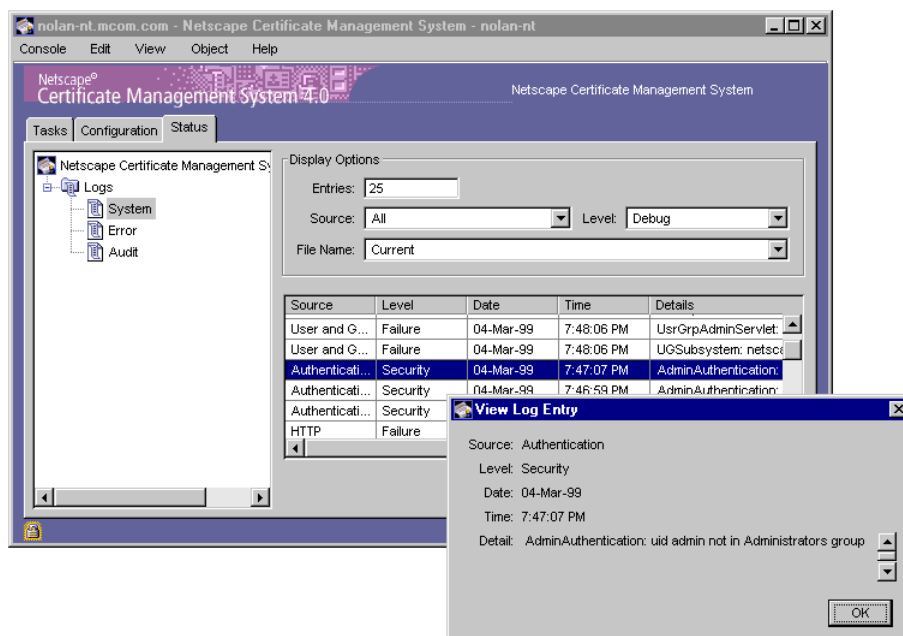
Certificate Management System maintains extensive system logs. These logs record various events and system errors for system monitoring and debugging. A system log records details such as the following:

- Each HTTP access invoked on the server
- Errors encountered, such as authentication failures, malformed universal resource indicators (URIs), invalid database password indications, and server start-up and shut-down messages
- Messages related to the status of certificate issuance or revocation, authentication failures for issuing-agent connections, and any errors related to the formatting of requests

You can view the contents of currently active as well as rotated system log files from the CMS window (see Figure 23.3).

If you have installed Certificate Management System on a Windows NT system, you can configure the server to log messages to Windows NT event log. For details, see “Logging to Windows NT Event Log” “Logging to Windows NT Event Log” on page 594.

Figure 23.3 A sample active system log displayed in the CMS window



To view the contents of an active or rotated system log file:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Status tab.
3. In the navigation tree, under Logs, click System.
4. In the Display Options section, specify your viewing preferences:

**Entries.** Type the maximum number of entries to be displayed. When this limit is reached, Certificate Management System returns any entries it has located that match the search request. If you enter zero (0), no messages are returned. If you leave the field blank, the server returns every matching entry (no limit) regardless of the number found.

**Source.** Select the CMS component (or service) for which log messages are to be displayed. Depending on the components that write to this log file, the drop-down list shows one or more of the following: All, Registration Authority, Certificate Authority, Key Recovery Authority, HTTP, Internal

Database, Authentication, Administration, LDAP, Request Queue, ACLs, User and Group, and Others. If you choose All, messages logged by all components that log to this file are displayed. For more information, see “Services That Are Logged” on page 569.

**Level.** Select a message category that represents the log level for filtering messages. For more information on log levels, see “Log Levels (Message Categories)” on page 570.

**Filename.** Select the log file you want to view. Choose Current to view the currently active system log file. For more information, see “Log File Naming Conventions” on page 573.

5. Click Refresh.

The table displays the system log entries. The entries are in reverse chronological order, with the most current entry placed at the top. Use the scroll arrows on the right edge of the panel to scroll through the log entries.

For each entry you see the following details:

**Source.** Indicates the CMS component or resource that logged the message.

**Level.** Indicates the severity of the corresponding entry (explained Table 22.3 on page 571).

**Date.** Indicates the date on which the entry was logged.

**Time.** Indicates the time at which the entry was logged.

**Details.** Provides a brief description of the log.

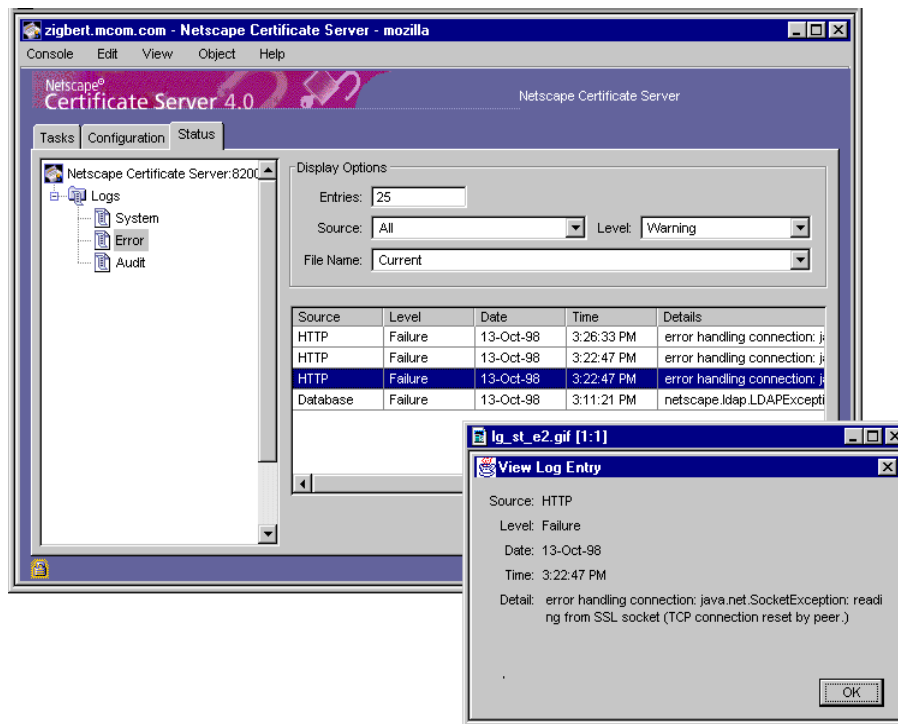
6. To view an entry in its entirety, either double-click it or select the entry and click View.

## Monitoring Error Logs

The error log file contains errors the server has encountered since the log file was created; it also contains informational messages about the server, such as when the server was started. Incorrect user authentication is also recorded in the error log. Use the error log to find broken URL paths or missing files.

You can view the contents of currently active as well as rotated error log files from the CMS window (see Figure 23.4).

Figure 23.4 A sample active error log displayed in the CMS window



To view the contents of an active or rotated error log file:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Status tab.
3. In the navigation tree, under Logs, click Error.
4. In the Display Options section, specify your viewing preferences:

**Entries.** Type the maximum number of entries to be displayed. When this limit is reached, Certificate Management System returns any entries it has located that match the search request. If you enter zero (0), no messages are returned. If you leave the field blank, the server returns every matching entry (no limit) to the client regardless of the number found.

**Source.** Select the CMS component (or services) for which log messages are to be displayed. Depending on the components that write to this log file, the drop-down list shows one or more of the following: All, Registration Authority, Certificate Authority, Key Recovery Authority, HTTP, Internal Database, Authentication, Administration, LDAP, Request Queue, ACLs, User and Group, and Others. If you choose All, messages logged by all components that log to this file are displayed. For more information, see “Services That Are Logged” on page 569.

**Level.** Select a message category that represents the level of logging to filter messages. For more information, see “Log Levels (Message Categories)” on page 570.

**Filename.** Select the log file you want to view. Choose Current to view the currently active error log file. For more information, see “Log File Naming Conventions” on page 573.

5. Click Refresh.

The table displays the error log entries. The entries are in reverse chronological order, with the most current log placed at the top. Use the scroll arrows on the right edge of the panel to scroll through the log entries.

For each entry you see the following details:

**Source.** Indicates CMS component or resource that logged the message.

**Level.** Indicates the severity of the corresponding entry (explained in Table 22.3 on page 571).

**Date.** Indicates the date on which the entry was logged.

**Time.** Indicates the time at which the entry was logged.

**Details.** Provides a brief description of the log.

6. To view an entry in its entirety, either double-click it or select the entry and click View.

# Monitoring Audit Logs

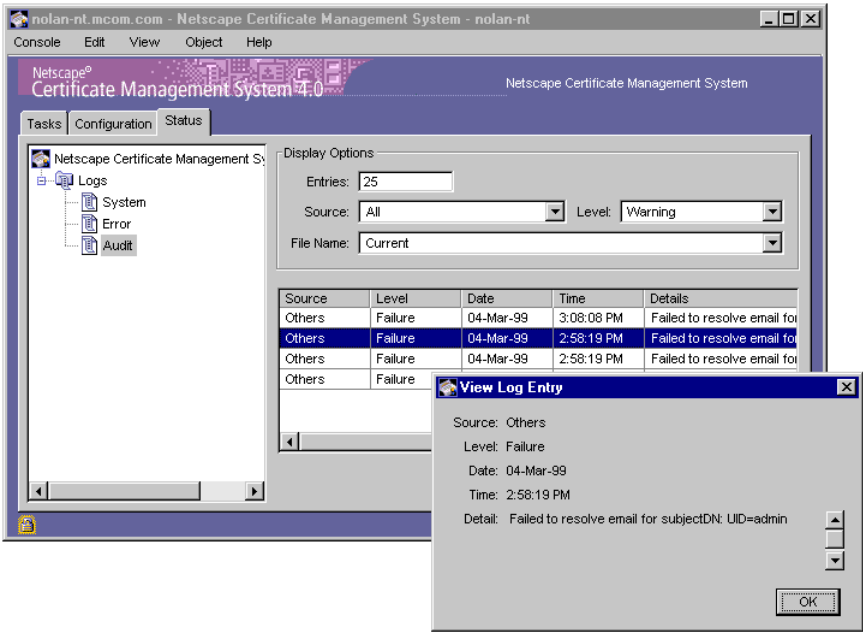
Certificate Management System maintains audit trails for all events—certificate requests, certificate renewal and revocation requests, CRL publication, and so on. These trails enable you to detect any unauthorized access or activity. The audit trails are logged and maintained in a file in your local file system.

If you have installed Certificate Management System on a Windows NT system, you can also configure the server to log audit messages to Windows NT event log. For details, see “Logging to Windows NT Event Log” on page 594.

**Important** You should periodically examine and audit the CMS audit log for unusual activity. When examining the log, note in particular the log entries that fall under the Security-Related Events category (these are labeled *Security*).

You can view the contents of currently active as well as rotated audit log files from the CMS window (see Figure 23.5).

Figure 23.5 A sample active audit log displayed in the CMS window





To view the contents of an active or rotated audit log file:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Status tab.
3. In the navigation tree, under Logs, click Audit.
4. In the Display Options section, specify your viewing preferences:

**Entries.** Type the maximum number of entries to be displayed. When this limit is reached, Certificate Management System returns any entries it has located that match the search request. If you enter zero (0), no messages are returned. If you leave the field blank, the server returns every matching entry (no limit) regardless of the number it finds.

**Source.** Select the CMS component (or resource) for which log messages are to be displayed. Depending on the components that write to this log file, the drop-down list shows one or more of the following: All, Registration Authority, Certificate Authority, Key Recovery Authority, HTTP, Internal Database, Authentication, Administration, LDAP, Request Queue, ACLs, User and Group, and Others. If you choose All, messages logged by all components that log to this file are displayed. For more information, see “Services That Are Logged” on page 569.

**Level.** Select a message category that represents the level of logging to filter messages. For more information, see “Log Levels (Message Categories)” on page 570.

**Filename.** Select the log file you want to view. Choose Current to view the currently active audit log file. For more information, see “Log File Naming Conventions” on page 573.

5. Click Refresh.

The table displays the audit log entries. The entries are in reverse chronological order, with the most current log placed at the top. Use the scroll arrows on the right edge of the panel to scroll through the log entries.

For each entry you see the following details:

**Source.** Indicates the CMS component or resource that wrote to the log file.

**Level.** Indicates the severity of the corresponding entry (explained in Table 22.3 on page 571).

**Date.** Indicates the date on which this entry was logged.

**Time.** Indicates the time at which this entry was logged.

**Details.** Provides a brief description of the log.

- 6. To view an entry in its entirety, either double-click it or select the entry and then click View.

# Using System Tools for Monitoring the Server (Windows NT Only)

If you have installed Certificate Management System on a Windows NT system, you can monitor the server with the system tools provided by Windows NT.

## Logging to Windows NT Event Log

You can also configure Certificate Management System to write both audit and system logs to the event log of a Windows NT system. The server's configuration file includes parameters that enable you to turn this feature on or off and to specify the levels for logging. The parameters are listed in Table 23.1:

Table 23.1 Configuration parameters for logging to the Windows NT event log

Configuration parameter	Description
<code>logNTAudit.NTEventSourceName</code>	Specifies the CMS instance ID for which the audit messages are to be logged. For example, the instance ID could be <code>cert-test CA</code> .
<code>logNTAudit.level</code>	Specifies the level for logging the audit messages. By default, it is set to 2. For information on log levels you can specify, see "Log Levels (Message Categories)" on page 570.

Table 23.1 Configuration parameters for logging to the Windows NT event log (Continued)

Configuration parameter	Description
<code>logNTAudit.on</code>	Specifies whether audit logging is enabled or disabled. To enable logging, enter <code>true</code> ; to disable logging, enter <code>false</code> . By default, it is enabled.
<code>logNTSystem.NTEventSourceName</code>	Specifies the CMS instance ID for which the system messages are to be logged. For example, the instance ID could be <code>cert-test CA</code> .
<code>logNTSystem.level</code>	Specifies the level for logging the system messages. By default, it is set to 2. For information on log levels you can specify, see “Log Levels (Message Categories)” on page 570.
<code>logNTSystem.on</code>	Specifies whether system logging is enabled or disabled. To enable logging, enter <code>true</code> ; to disable logging, enter <code>false</code> . By default, it is enabled.

Note that by default both the audit and system logs are enabled and the log levels for both is set to 2. You can change this by editing the configuration file (no UI is provided for this).

To change the default configuration:

1. Stop the CMS instance; see “Stopping Certificate Management System” on page 110.
2. Open the configuration file (`CMS.cfg`) in a text editor; to locate the file, see “Locating the Configuration File” on page 71.
3. Locate the configuration parameters pertaining to Windows NT logging, and change the parameter values as appropriate. The sample configuration below shows the audit log turned on with a logging level of 1. The system log is turned off.

```
logNTAudit.NTEventSourceName=cert-testCA
logNTAudit.level=1
logNTAudit.on=true
logNTSystem.NTEventSourceName=cert-testCA
logNTSystem.level=2
logNTSystem.on=false
```

4. Save your changes, and close the configuration file.
5. Start the CMS instance; see “Starting Certificate Management System” on page 106.

## Using Event Viewer

In addition to logging messages to the log files maintained in your local file system, Certificate Management System can also log audit messages and system errors to the Windows NT Event log. To configure the server to log messages to Windows NT event log, see “Logging to Windows NT Event Log” on page 594. If you configure the server to do so, you can use the system’s tool called *Event Viewer* to monitor events related to your server.

More information about Event Viewer is available in your system documentation.

To monitor Certificate Management System by using Event Viewer:

1. In the Administrative Tools program group, double-click the Event Viewer icon.
2. From the Log menu, select Application.

The Application log appears in Event Viewer. In this log, the *source* of any messages from Netscape Certificate Management System is the server’s instance ID (if you didn’t change the default values assigned to the `logNTAudit.NTEventSourceName` and `logNTSystem.NTEventSourceName` parameters).

3. From the View menu, choose Find to search for one of the Netscape labels in the log; use Refresh to see updated log entries.

4. Double-click a log entry to see additional information.

The mapping between the CMS log levels and the Windows NT event type is shown in Table 23.2.

**Table 23.2 Mapping between CMS log levels and Windows NT event log type**

Windows NT log event type	CMS log level
Information	Debugging (0)
Information	Informational (1)
Warning	Warning (2)
Error	Failure (3)
Error	Misconfiguration (4)
Error	Catastrophic failure (5)
Error	Security-related events (6)

## Signing Log Files

Certificate Management System allows you to digitally sign log files before you archive them or distribute them for audit purposes. This feature enables you to check whether the log files have been tampered with since being signed.

For signing log files, you use a command-line utility called *Netscape Signing Tool*; for details about this utility, see Appendix F, “Netscape Signing Tool” in the online version of this document; to locate the online version of the document, see “Where to Go for Related Information” on page 27. The utility uses information in the certificate (`cert7.db`), key (`key3.db`), and security module (`secmod.db`) databases of Certificate Management System.

Before you begin signing the log files, follow these guidelines:

- Determine the key pair you want to use for signing the log directory. Typically, you should use the Certificate Manager’s (the CA’s) signing key pair. Also find out the nickname of the certificate that corresponds to this key pair.
- If you have deployed many CAs, locate the CMS instance in which the CA you want to use is installed.

- Find out whether the key pair is in an internal or external token. If it is in an external token, make sure the token is currently installed. You will also need to know the password for the token.
- Determine which log files need to be signed. Put all the files that need to be signed in one or more directories. (The utility can sign a directory containing files; it cannot sign individual files.) Make sure these directories are in the host machine in which the CA is installed.
- Determine names for the output files; the output you receive will be a JAR file (which is a signed zip file). You may want to give names that will help you identify these JAR files easily in the future.

When you are ready with all this information, follow the procedure below to sign the log directories:

1. Go to the CMS instance in which the CA whose key pair you want to use for signing is installed.
2. Copy the security module database (`secmod.db` file) from the Administration Server configuration directory to the CMS configuration directory.

The security module database is at this location:

```
<server_root>/admin-serv/config/...
```

Copy it to this location:

```
<server_root>/cert-<instance_id>/config/...
```

3. Open a terminal window.

4. At the command prompt, run the following command with the appropriate information:

```
signtool -d <secdb_dir> -k <cert_nickname> -Z <output>
<input>
```

<secdb\_dir> specifies the path to the directory that contains the certificate, key, and security module databases for the CA. This must be the same path you used to copy the security module database in step 2.

<cert\_nickname> specifies the nickname of the certificate you want the utility to use for signing.

<output> specifies the name of the JAR file (a signed zip file).

<input> specifies the path to the directory that contains the log files.

For example, in a Windows NT system, your command might look like this:

```
signtool -d c:\netscape\server4\cert-testCA\config -k
testCAsigningcertificate -Z log_err_02_99.jar
c:\archive\logs
```

where

c:\netscape\server4\cert-testCA\config is the path to the certificate, key, and security module databases (secdb\_dir).

testCAsigningcertificate is the certificate nickname (cert\_nickname).

log\_err\_02\_99.jar is the name of the JAR file (output).

(input) is c:\archive\logs is the directory to be signed.





# *Issuance and Management of End-Entity Certificates*

*Chapter 24* Issuing and Managing End-Entity Certificates

*Chapter 25* Recovering Encrypted Data



# Issuing and Managing End-Entity Certificates

This chapter explains how Netscape Certificate Management System (CMS) issues and manages end-entity certificates.

The chapter has the following sections:

- Certificate Issuance to Servers (page 603)
- Certificate Issuance to Routers (page 613)
- Certificate Renewal (page 619)
- Certificate Revocation (page 621)

## Certificate Issuance to Servers

For Certificate Management System to generate a server certificate, it must receive the certificate signing request (CSR) from the server that needs the certificate. This request must be initiated by the administrator of the specific server requiring the certificate.

SSL-enabled servers (or servers that are capable of using certificates for security) provide mechanisms for generating a CSR based on new or existing key pairs. For example, servers that belong to the Netscape's version 4.x server family come with a wizard that walks an administrator through the entire

process of requesting a server certificate and installing it in the server's certificate database. For information on this wizard, see "Obtaining and Installing a Certificate" in *Managing Servers with Netscape Console*.

Once an administrator generates a CSR for a server, he or she must paste it into the appropriate server enrollment form hosted by a Registration Manager or Certificate Manager, and then submit the request. Upon receipt of the request, Certificate Management System responds as follows:

1. Verifies the validity and authenticity of the request.

The authentication mechanism that Certificate Management System uses is based on the authentication mechanism specified in the enrollment form the administrator uses to submit the certificate request. For example, if the enrollment form was configured to employ directory-based authentication, Certificate Management System checks the configured directory for the appropriate information. On the other hand, if the enrollment form specifies manual authentication, the request gets queued and awaits approval by an agent.

2. Subjects the request to policy checks.

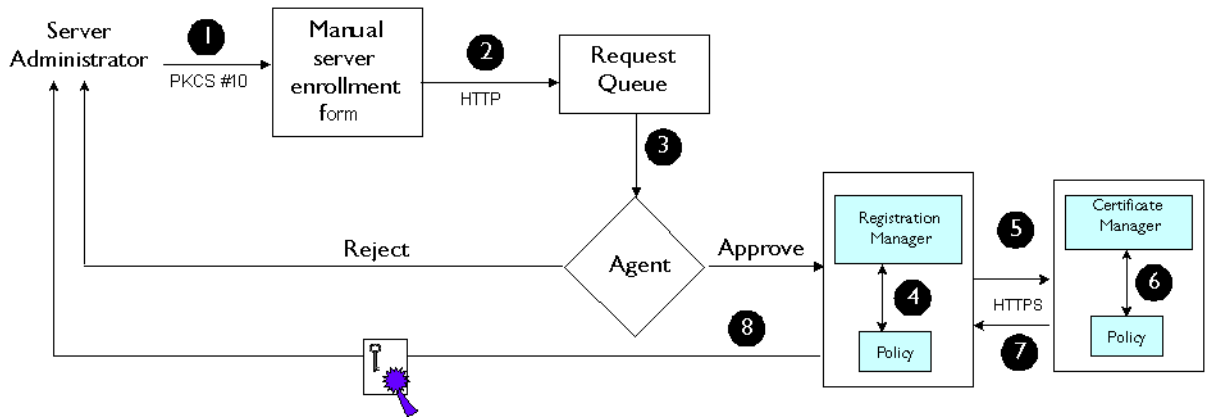
If the request passes all the policy rules, Certificate Management System generates the server certificate and sends it to the email address specified in the server certificate request (the enrollment form includes a field for the administrator to enter this information). Otherwise, Certificate Management System logs an error message.

Upon receipt of the certificate, the server administrator installs the certificate in the server's certificate database.

## How the Manual Server Enrollment Process Works

Figure 24.1 illustrates how Certificate Management System issues a server certificate in a deployment scenario involving a Registration Manager acting as an enrollment authority to a Certificate Manager. The server certificate is requested via a manual enrollment form hosted by the Registration Manager.

Figure 24.1 Server (or site) certificate issuance



These are the steps shown in Figure 24.1:

1. The server administrator goes to the manual enrollment form hosted by the Registration Manager, pastes in the certificate signing request in PKCS #10 format, completes the other information in the enrollment form, and submits the form.

(If the enrollment port is HTTPS, the administrator should visit the link that delivers the CA's certificate chain and download the chain into the browser that he or she will use for server enrollment.)

2. The Registration Manager verifies the authenticity of the request. Because the request requires manual authentication, the Registration Manager stores the request in the queue for agent approval.
3. An agent processes the request and either rejects or approves it.
4. The Registration Manager picks up the approved request and subjects it to policy checks.
5. If the request passes the Registration Manager's policy checking, the Registration Manager submits the request to the Certificate Manager for signing. The Certificate Manager verifies the authenticity of the Registration Manager by verifying the certificate presented by it. If it is a trusted Registration Manager, the Certificate Manager accepts the request.

6. The Certificate Manager subjects the request to its own policy checks.
7. If the request passes Certificate Manager's policy, it signs the request immediately and returns the certificate to the Registration Manager. The Registration Manager then delivers the certificate to the administrator. Optionally, the Certificate Manager may publish the certificate to the corporate directory.

If the Certificate Manager's policy requires additional information, the administrator will be directed to return later to pick up the certificate. The administrator may need to query the Registration Manager using the certificate request number to see whether the certificate has been issued. Alternatively, the Registration Manager can be configured to email the user when the certificate is ready for pick up. See "Notifications of Certificate Issuance to End Entities" on page 365.

8. The Registration Manager delivers the server SSL certificate to the email address specified in the enrollment form. Optionally, the Registration Manager may publish the certificate to the corporate directory.

## Getting Server SSL Certificates for Netscape Servers

To enable a server to establish SSL connections, you need to get a certificate that identifies the server. You can get a certificate for a server by submitting a request to Certificate Management System.

To generate the actual request, you (or the server administrator) need to use the server that requires the certificate. This is required because the private key must be stored with the server that will use it.

The following section explains how to request a server SSL certificate for Netscape servers. The instructions apply mainly to requests from servers other than CMS subsystem server—for example, Netscape Enterprise, Administration, and Directory Servers. To request a certificate for a CMS subsystem, follow the instructions in "Getting New Certificates for the Subsystems" on page 232.

## Getting Certificates for Version 3.x Servers

To get a certificate for a server in the Netscape version 3.x server family (for example, Netscape Administration Server 3.x) follow the procedure below:

- Step 1. Generate a Server Certificate Request for Your Server
- Step 2. Submit the Server Certificate Request to Certificate Management System
- Step 3. Install Your Server's SSL Certificate
- Step 4. Accept a CA as Trusted in Your Server
- Step 5. Verify Your Server's SSL and CA Certificates

### Step 1. Generate a Server Certificate Request for Your Server

To generate the certificate signing request for a server:

1. Open a web browser window.
2. Go to the Administration Server, and use the Server Selector to access the Server Manager for your server.
3. Follow the directions presented there to generate a new key pair which you will then get certified (you will use this key pair to generate a certificate signing request).

Alternatively, you can use any other tool provided with your server to generate the key pair; see the documentation for your server.

4. Once you have generated a key pair, follow the directions presented to generate a certificate signing request (CSR).
5. In the Certificate Authority field, enter your own email address.

The server mails the request to the address specified in this field.

6. Submit the form.

The server generates and displays a CSR.

7. Copy the CSR, including the -----BEGIN NEW CERTIFICATE REQUEST----- and -----END NEW CERTIFICATE REQUEST----- marker lines, to a text file. For example:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBBzCBsgIBADBPMQswCQYDVQQGEwJVUzEoMCYGA1UEChMfTmV0c2NhGUGRGlYzWN0b3J5IFB1Y
mxpY2F0aW9uczEWMBQGA1UEAxMNZHVtY29tLmNvbTBaMA0GCSqGSIb3DQEBAQU2nfjiMEYCQQ
CksMRALGdfp4m0OiGcgiJG5KgOsyRNvWGYW7kfW+8mmiJdtZRjYNjjcgPF3VnlsbxbclX9LVjjNLC
57u37XZdAgEDoAAwDQYJKoZIhvcNAQEBQADQCCYUtnUtCVGyNrYGSfydc1qiovxy1fRD1z23zg+e
BPK7n85UyE4r5zGZjDsMYr172ytfAFL7DeG83DWzr8Z
-----END NEW CERTIFICATE REQUEST-----
```

Next, you need to paste this request into the server enrollment form hosted by Certificate Management System.

## Step 2. Submit the Server Certificate Request to Certificate Management System

To submit the server certificate request to Certificate Management System:

1. Open a web browser.
2. Go to the server enrollment form (the page that allows you to submit a server certificate request).

By default, the enrollment forms are at this location:

```
https://<host_name>:<end_entity_HTTPS_port> or
http://<host_name>:<end_entity_HTTP_port>
```

3. Select the form that you want to use.

By default, Certificate Management System provides forms that employ manual and directory-based authentication. For information on how these forms work, see “How the Manual Server Enrollment Process Works” on page 604 and “Getting Server SSL Certificates for Netscape Servers” on page 606.



4. Complete the request form with the information that Certificate Management System needs to create a certificate for your server.

In general, you will be required to enter the following information:

- In the certificate request text area, paste the CSR that you copied to the text file, including the -----BEGIN NEW CERTIFICATE REQUEST----- and -----END NEW CERTIFICATE REQUEST----- marker lines.
- In the contact information section, enter values to identify yourself. These values will be used by the CA, if the need arises. For example, if there are any questions or problems with the certificate request, the CA administrator or agent will use this information to contact you. Also, be sure to enter your email address. This is the address where the CA will send the certificate once it has been issued.
- In the additional comments section, enter any additional information that might help the issuing agent process the request. For example, you might want to enter the name of the person who instructed you to obtain a certificate or some other administrative information.

5. Submit the request.

You should receive notification from Certificate Management System or an issuing agent (depending on which enrollment form you used) when your request is processed. The notification will contain your certificate, along with information on how to install the new certificate into your server. The notification may also mention that you need to install the CA's certificate as a trusted CA. Check the notification message for details.

### Step 3. Install Your Server's SSL Certificate

To install the server SSL certificate on your server:

1. Open a web browser window.
2. Go to the Administration Server, and use the Server Selector to access the Server Manager for your server.

3. Follow the directions presented there to install the certificate.

In general, you will be required to specify or enter the following information:

- Whether the certificate is for this server. Be sure to select the option that says the certificate is for this server.
  - A name (or nickname) for the certificate. This name will be displayed in the list of certificates installed on this server.
  - The certificate, in base-64 encoded format. Open the email sent to you by the CA, locate and copy the portion that begins with -----BEGIN CERTIFICATE----- and ends with -----END CERTIFICATE-----, and paste it into the text area in the form.
  - The encryption alias. Enter the alias for your server.
4. Follow the prompts and add the certificate to your server's certificate database.
  5. Stop and restart Administration Server for the changes to take effect.

The server decrypts the message, extracts the certificate, and saves it to the directory you specified.

### Step 4. Accept a CA as Trusted in Your Server

In both Netscape clients and servers, CAs can be either *trusted* or *untrusted*. If a CA is trusted, Netscape clients and servers accept the certificates that have been issued by that CA. For the server to accept (during SSL client authentication) client certificates that have been issued by Certificate Management System, you must import its certificate chain into the certificate database of your server.

To view this chain in a format that can be used by Netscape servers:

1. Go to the home page of Certificate Management System.

By default, the home page is at this location:

```
https://
<machine_name>.<your_domain>.<domain>:<end_entity_port>
```

2. Click Accept “This Authority in Your Server.”
3. Specify how you want Certificate Management System to display the certificate chain.

You can choose to display the entire certificate chain (in a single block) or individual certificates in the chain. The entire certificate chain is in PKCS #7 format. If you are using an older server that does not recognize the complete certificate chain format, you may need to display each individual certificate in the chain (for example, a version earlier than Netscape server 2.0 releases).

4. Specify how you want to trust this CA.

You can choose to trust only the CA you are accessing or all authorities whose certificates are included in the chain.

5. Click Present Certificate Chain.

If you chose to display the whole chain for importing into your server, the certificate chain is displayed in a format similar to this:

-----BEGIN CERTIFICATE-----

```
MIIBtgYJYIZIAYb4QgIFoIIBpzCAZ8wggGbMIIBRaADAgEAAgEBMA0GCSqGSIb3DQEBAUAMFcxC
zAJBgNVBAYTALVTMSwwKgYDVQQKEyNOZXRzY2FwZSBDb21tdW5pY2F0aW9ucyBDb3Jwb3JhdGlvbj
EaMBGGA1UECXMRSXNzdWluZyBBdXRob3JpdHkwHhcNOTYxMTA4MDkwNzM0WhcNOTg1MTA4MDkwNzM
0WjBXMQswCQYDVQQGEwJVUzEsMCoGA1UEChMjTmV0c2NhcnGUGuG29tbXVuaWNhdGlvbnMgQ29ycG9y
YXRpb24xGjAYBgNVBAsTEU1zc3VpbmcgQXV0aG9yaXR5MFowDQYJKoZIhvcNAQEBBQADSAwRgJBA
OBiQPcK8851jjQXA2GBsaKNFg6pYaM3qhQhM0w5EiY6P1ttMjc5M1PizZHd1gNdQLzaNoLMVKjOV5
sBp+fkCAQMwDQYJKoZIhvcNAQEEBQADQCWPU4gI5uaWM3EAbXfhQ
```

-----END CERTIFICATE-----

6. Open a new web browser window.
7. Go to the Administration Server, and use the Server Selector to access the Server Manager for your server.
8. Follow the directions presented there to install the certificate chain.

In general, you will be required to specify or enter the following information:

- Whether the certificate is for this server or a trusted CA. Be sure to select the option that says the certificate is for a trusted certificate authority (CA).

- A name (or nickname) for the certificate chain. This name will be displayed in the list of certificates installed on this server.
  - The certificate chain, in PKCS #7 format. In the original browser window (the window displaying the encoded certificate chain), copy the portion that begins with -----BEGIN CERTIFICATE----- and ends with -----END CERTIFICATE-----, and paste it into the message text area in the form.
9. Save your changes.
  10. Stop and restart your Administration Server.

### **Step 5. Verify Your Server's SSL and CA Certificates**

Before activating your server for SSL connections, you can verify whether you have installed your server's SSL and CA certificates correctly.

1. Open a web browser window.
2. Go to the Administration Server, and use the Server Selector to access the Server Manager for your server.
3. Follow the directions there to get to the area that allows you to manage your server's certificates.
4. Scroll to the bottom of the list to find the SSL and CA certificate chain you installed (identified by the nicknames you specified).

If you find both of them, your server is ready for SSL configuration. If not, you must go through the steps again to correctly install whichever certificate is missing.

## **Getting Certificates for Netscape Version 4.x Servers**

For Netscape version 4.x servers, you can use the Certificate Setup Wizard provided by Netscape Console to get new certificates, renew existing certificates, and install certificates in the database of a server. For information about this wizard, see to *Managing Servers with Netscape Console*.

# Certificate Issuance to Routers

Cisco routers support the use of certificates for authentication, encryption, and tamper detection by using the IP Security (IPSec) protocol. Certificate Management System supports Cisco's PKI protocol, the Certificate Enrollment Protocol (CEP); this protocol runs over HTTP and provides its own form of encryption. For an overview of certificate authority support for IPSec, see the information available at this URL:

[http://www.cisco.com/warp/public/778/security/821\\_pp.htm](http://www.cisco.com/warp/public/778/security/821_pp.htm)

You can issue certificates to routers using Certificate Management System. Routers use certificates to authenticate each other and to establish an encrypted IPSec channel between them; all TCP/IP communication passes through this encrypted channel.

In general, issuing a certificate to a router involves the following steps:

- Step 1. Find the Required Information
- Step 2. Generate the Key Pair for the Router
- Step 3. Request the CA's Certificate
- Step 4. Submit the Certificate Request to the CA

## Step 1. Find the Required Information

- Decide whether you want to submit the certificate request for your router to the Certificate Manager (CA) directly or through a Registration Manager.
- Open Netscape Console, and locate the CMS instance that corresponds to the subsystem of your interest. Make sure that the Certificate Manager is started. If you are planning to submit the request via a Registration Manager, make sure that both the Registration Manager and Certificate Manager (to which the Registration Manager forwards its requests) are started.
- Open the CMS window, and verify whether the HTTP port is enabled. If it isn't, enable it; see "Configuring Port Numbers" on page 125. If you are requesting the certificate for an earlier version of router software, make sure that the HTTP port number is set to 80; earlier versions of router software can only connect to port 80.

- Note the CEP enrollment URL. It is in this form:

`http://<host_name>:<HTTP_port>/cgi-bin/pkiclient.exe`

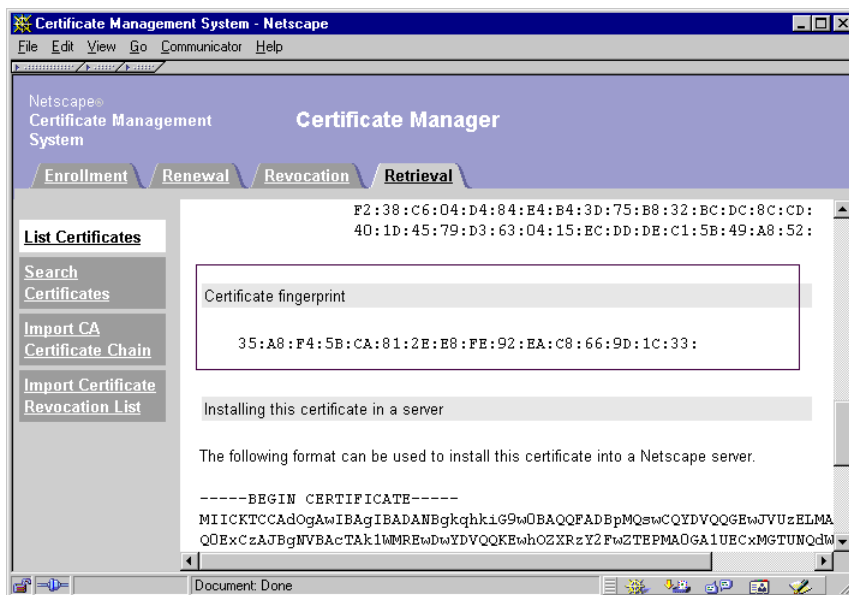
where <host\_name> is in the form

`<machine_name>.<your_domain>.<domain>`

- Note or print the certificate fingerprint information of the Certificate Manager *CA signing certificate*. You will be required to compare this with the fingerprint the router will show on the screen.

To locate the fingerprint information:

1. Go to the end-entity page hosted by the Certificate Manager.
2. Click the Retrieval tab.
3. List or search for the CA signing certificate.
4. Click Details.
5. Scroll down to the section that says “Certificate fingerprint.”



- In your router documentation, locate the information specific to requesting certificates for routers. Check the signing algorithm, such as RSA or DSA, and key lengths, such as 512 and 1024, supported by the router. Based on that information, determine the signing algorithm and the key length for the certificate you want to request.
- Find out the password that enables you to access the router in *privileged* mode.
- In your router documentation, locate instructions for requesting certificates. You will be required to run the appropriate commands using this documentation.

## Step 2. Generate the Key Pair for the Router

Run the appropriate commands for your router, and generate the key pair. You will be required to provide the signing algorithm, such as RSA or DSA, and the key length, such as 512 or 1024. The longer the key length, the more time the router takes to generate the key pair.

## Step 3. Request the CA's Certificate

In this part of the operation, you identify the CA to the router, thus enabling the router to authenticate the CA from which it will request the certificate. You also verify whether the router is talking to the right CA; you do this manually.

Here's what you should do:

1. Run the appropriate command to get the CA certificate.

The command will ask you to specify the following:

- An identity for the CA. You can give any identity; choose something you will remember, since you will be required to provide it when you submit the certificate request.
  - The CA's enrollment URL; this is the enrollment URL you identified in Step 1.
2. The router gets the CA certificate and displays its fingerprint on your screen.

3. Verify the fingerprint on your screen with the one you noted down in Step 1.

If it matches, the router is talking to the right CA.

## Step 4. Submit the Certificate Request to the CA

To submit the certificate request to the CA:

1. Run the appropriate command.

The command will ask you for certain information:

- The CA's identity. You specified this in Step 3.
- Challenge password. If you enter one, write it down; you will be required to specify this password to revoke the certificate.
- The CEP enrollment URL.
- Whether you want to include the router's serial number in the request. If you choose to include the serial number, it will be included in the certificate's subject name.
- Whether you want to include the router's IP address in the request. If you choose to include the IP address, it will be included in the certificate's subject name.

2. This step depends on your CA's configuration for router enrollment.
  - If the CA to which the router submitted the request employs automatic enrollment (or authentication) for routers, the request will get processed by the CA. The CA may return the certificate to the router in the same transaction. If it doesn't, the router checks with the CA at periodic intervals; in the router configuration you can specify how often the router should poll the CA for the certificate and how many attempts it should make. By default, the router checks the CA every minute.
  - If the CA to which you submitted the request is configured for manual enrollment (or authentication), the request gets queued and awaits approval by an agent.



**Important** Your router may require additional configuration changes. Be sure to follow the information in your router documentation.

## Example

The example below shows the commands and associated outputs for a Cisco router:

```
# To perform certificate enrollment for a router using CEP, you must be
# in privileged mode, which you do by typing "enable" first, and then
# entering the password.
```

```
router> enable
```

```
router% config terminal
```

```
router(config)#crypto key generate rsa
```

```
The name for the keys will be: netscape.mcom.com
```

```
Choose the size of the key modulus in the range of 360 to 2048 for
your General Purpose Keys. Choosing a key modulus greater than 512
may take a few minutes.
```

```
How many bits in the modulus [512]:
```

```
Generating RSA keys ...
```

```
[OK]
```

```
router(config)#crypto ca identity test-ca
```

```
router(ca-identity)#enrollment url http://ca-hostname.domain.com/
cgi-bin/pkiclient.exe
```

```
router(ca-identity)#exit
```

```
router(config)#crypto ca authenticate test-ca
```

```
Certificate has the following attributes:
```

```
Fingerprint: 24D34656 EB830C39 DD9E8179 0A4EBA98
```

```
% Do you accept this certificate? [yes/no]: yes
```

```
router(config)#crypto ca enroll test-ca
```

```
%
```

```
% Start certificate enrollment ..

% Create a challenge password. You will need to verbally provide this
password to the CA Administrator in order to revoke your
certificate. For security reasons your password will not be saved
in the configuration. Please make a note of it.

Password:

Re-enter password:

% The subject name in the certificate will be: router.domain.com
% Include the router serial number in the subject name? [yes/no]: yes
% The serial number in the certificate will be: 08342063
% Include an IP address in the subject name? [yes/no]: yes
Interface: ethernet0
Request certificate from CA? [yes/no]: yes
% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto ca certificate' command will also show the
fingerprint.

router(config)# exit

router#show crypto ca certificates

CA Certificate

Status: Available

Certificate Serial Number: 1

Key Usage: Not Set

Certificate

Subject Name

Name: netscape.mcom.com

IP Address: 208.12.63.193

Serial Number: 08342063

Status: Pending
```

Key Usage: General Purpose

Fingerprint: 91D70D7F D8BF0DFA E13F00B0 6EA706A0 00000000

## Certificate Renewal

Every certificate issued by Certificate Management System has a validity period that determines its expiration date. If you have configured the Certificate Manager or Registration Manager to use the *validity constraints* policy (see “Validity Constraints Policy” on page 426), the validity period of a certificate is determined by the policy settings at the time the certificate was issued. For a certificate to be valid beyond its expiration date, it must be renewed. Otherwise, the certificate becomes invalid, and the entity owning the certificate will no longer be able to use it. Also, the expired certificate will take up space in your publishing directory and in the internal database of Certificate Management System.

**Note** The Job scheduler component of Certificate Management System enables you to schedule a *job* for removing expired certificates from the publishing directory. For details, see “Directory Update and Notification” on page 358.

## Renewal of Client Certificates

Certificate Management System allows end users to renew their certificates by using the certificate renewal form hosted by a Certificate Manager or Registration Manager. End entities can use this form for renewing a single certificate or dual certificates. To renew a certificate, end entities need to present their current certificate (for SSL client authentication) during renewal. For renewal purposes, the server accepts even expired certificates for verifying the end entity.

Certificate Management System comes with a policy plug-in module that allows you to configure both Certificate Manager and Registration Manager to apply specific renewal rules for certificates; you can specify how long a renewed certificates should be valid and how many days before expiration of the current certificate an end entity can renew the certificate. The renewal policy rule, if enabled, also enforces that the certificate presented during client authentication

is valid or is expired; it cannot have been revoked. The For more information about this policy module, see “Renewal Validity Constraints Policy” on page 419.

When a certificate is renewed, Certificate Management System formulates a new certificate with the same public key and other details from the existing certificate; the renewal does not include key changeover. The server, if configured for LDAP publishing, also publishes the new certificate to the publishing directory.

The following steps describe how a Certificate Manager or Registration Manager renews client certificates:

1. If renewal is done automatically by the life-cycle management module, the server fetches end entities' certificates from its certificate repository, the internal database. If an end entity uses the renewal form, the end entity can request renewal of the authentication certificate only or renewal of all certificates that belong to the end entity with the same subject name as in the authentication certificate. After successful authentication, the server fetches the currently valid certificates for the end entity from its certificate repository.
2. The server formulates a renewal request based on the certificate being used for renewal.
3. The server applies its currently configured renewal policies to the request. If the policies require that the request be deferred for agent approval, the server stores the renewal request in the request queue and responds to that end entity indicating that the request is waiting for approval by an agent. If the request is rejected by policy modules, the server sends an error response.
4. The server formulates a CRMF request for one or more certificates as the case may be using a suitable template.

As an administrator, you can configure a Certificate Manager or Registration Manager to automatically notify end entities to renew their certificates before the current ones expire. If you enable this feature, the subsystem periodically queries the internal database for *already expired* or *about to expire and not renewed already* certificates, and alerts the user and you (optional) to imminent certificate expiration. For more information about this feature, see “Certificate Renewal Notifications” on page 347.

## Renewal of Server Certificates

Certificate Management System allows server administrators to renew their certificates by using the server enrollment form hosted by a Certificate Manager or Registration Manager. The renewal process is similar to the enrollment process in that the administrators must manually generate the certificate signing request using the server's key pair, paste that request in the manual enrollment form, and submit the request. For details, see "Certificate Issuance to Servers" on page 603.

**Note** For renewing the certificates of a Certificate Manager, Registration Manager, or Data Recovery Manager, see "Renewing Certificates for the Subsystems" on page 239.

## Certificate Revocation

Certificate Management System allows a certificate to be revoked by an end entity (the original owner of the certificate) or by a Certificate Manager or Registration Manager agent. End entities can revoke certificates by using the Revocation form provided in the end-entity services interface. Agents can revoke end-entity certificates by using the appropriate form in the Agent Services interface. Certificate-based authentication (SSL client authentication) is required in both cases.

- An end entity can revoke only those certificates that contain the same subject name as in the certificate presented for authentication. After successful authentication, the server lists the certificates belonging to the end entity. The end entity can then select the certificates to be revoked or can revoke all certificates in the list. The end entity can also specify additional details, such as the date of revocation and revocation reason for each certificate or for the list as a whole. For instructions on how end entities revoke their certificates, see the online help associated with the end-entity forms.
- Agents can revoke certificates based on a range of serial numbers or based on one or more subject name components. Upon submission of the revocation request, the agent receives a list of certificates from which she or he can pick the ones to be revoked. For instructions on how agents revoke end-entity certificates, see *Netscape Certificate Management System Agent's Guide*.

Upon receiving the list of certificates to be revoked, the Registration Manager formulates a CMMF request and sends it to the Certificate Manager. The Certificate Manager marks the corresponding certificate records in its certificate store (maintained in the internal database) as *revoked* and if configured to do so, removes the revoked certificates from the publishing directory and updates the CRL in the publishing directory.

# Recovering Encrypted Data

When data is stored in encrypted form, you must have the private key that corresponds to the public key that was used to encrypt the data in order to decrypt and read it. If the private key is lost, the data cannot be retrieved. A private key can be lost because of a hardware failure, for example, or because the key's owner forgets the password or loses the hardware token in which the key is stored. Similarly, encrypted data cannot be retrieved if the owner of the key is unavailable to supply it—for example, has left the organization that owns the data.

This chapter explains how to use the Data Recovery Manager to archive users' encryption private keys and how to use the archived keys later, in place of missing encryption keys, to recover encrypted data.

The chapter has the following sections:

- PKI Setup for Key Archival and Recovery (page 624)
- Key Archival Process (page 626)
- Key Recovery Process (page 629)
- Setting Up Key Archival and Recovery Process (page 641)

# PKI Setup for Key Archival and Recovery

To be able to archive users' encryption private keys and recover them later, you need a PKI setup that includes the following elements:

- Clients that can generate dual keys and that support the key archival option (using the CRMF/CMMF protocol)
- An installed and configured Data Recovery Manager
- HTML forms with which your users can request dual certificates (based on dual keys) and key recovery agents can request key recovery

The sections that follow explain these elements in detail. For step-by-step instructions on setting up your PKI environment for key archival and recovery, see "Setting Up Key Archival and Recovery Process" on page 641.

## Clients That Can Generate Dual Key Pairs

Only keys that are used exclusively for encrypting data should be archived; signing keys in particular should never be archived. Having two copies of a signing key would defeat the certainty with which the key identifies its owner; a second copy could be used to impersonate the digital identity of the original key owner.

Clients that generate single key pairs use the same private key for both signing and encrypting data, so you cannot archive and recover a private key deriving from a single key pair. By contrast, clients that can generate dual key pairs use one private key for encrypting data and the other for signing data. Because the encryption private key is separate, you can archive it.

In addition to generating dual key pairs, your users' clients must also support the encryption key archival option in certificate requests. This option triggers the key archival process at the time encryption private keys are generated as a part of certificate issuance.

Future versions of Netscape Communicator support both dual key-pair generation and the key archival option.



## Data Recovery Manager

With the Data Recovery Manager, you can archive data encryption keys when they are created during dual key-pair generation. You can then recover the keys if they are lost or the key owner is unavailable.

The Data Recovery Manager can archive and recover keys only from clients that support dual key-pair generation and the key archival option in certificate requests.

Certificate Management System does not provide any policy plug-in modules for the Data Recovery Manager. However, you can write custom policy plug-in modules (that is, write Java classes that implement these rules), register them in the Data Recovery Manager's policy framework, and create policy rules using these plug-in implementations.

## Forms for Users and Key Recovery Agents

End users' encryption private keys are archived by the Data Recovery Manager when they are generated. So, for key archival to occur, the enrollment form that users fill out to request dual certificates must have the JavaScript code for activating the key archival process embedded in it, along with a valid copy of the Data Recovery Manager's transport certificate. Then, when a Certificate Manager or Registration Manager that is processing the user's certificate issuance request detects the key archival option, it automatically requests the service of the Data Recovery Manager. For information on customizing this form, see "Step 3. Customize the Certificate Enrollment Form" on page 643.

Initiating the key recovery process also requires its own HTML form. By default, the Data Recovery Manager Agent Services interface provides a form for initiating the process and retrieving keys. For information on customizing this form, see "Step 4. Customize the Key Recovery Form" on page 653.

## Key Archival Process

If your certificate infrastructure has been set up for key archival, the Data Recovery Manager automatically archives users' encryption private keys. For general information on the type of PKI setup needed for archiving keys, see "PKI Setup for Key Archival and Recovery" on page 624. For specific instructions on setting up a key archival and recovery infrastructure, see "Setting Up Key Archival and Recovery Process" on page 641.

## Why You Should Archive Keys

If a user loses a private data-encryption key or is unavailable to use his or her private key, the key must be recovered before any data that was encrypted with the corresponding public key can be read. You can recover the private key if an archival copy of it was created when the key was generated.

Here are a few situations in which you might need to recover a user's encryption private key:

- An employee loses the encryption private key (for example, after a disk crash or by forgetting the password to the key file) and cannot read encrypted mail messages.
- An employee is on an extended leave, and you need access to an encrypted document in his or her files.
- An employee leaves the company, and company officials need to perform an audit that requires gaining access to the employee's encrypted mail.

## Where the Keys are Stored

If configured properly, the Data Recovery Manager stores your users' encryption private keys automatically whenever the associated or connected Registration Manager or Certificate Manager issues certificates to your users. The Data Recovery Manager stores encryption private keys in a secure key repository in its internal database; each key is stored as a key record.

The archived copy of the key remains encrypted (or wrapped) with the Data Recovery Manager's storage key; see "Storage Key Pair" on page 191. It can be decrypted (or unwrapped) only by using the corresponding private key, to which no individual has direct access. A combination of one or more key recovery agents' passwords enables the Data Recovery Manager to retrieve its private storage key and use it to decrypt and recover an archived key. For details on how this process works, see "Key Recovery Agents and Their Passwords" on page 630.

The Data Recovery Manager indexes stored keys by key number (or ID), owner name, and a hash of the public key, allowing for highly efficient searching by name or by public key. The key recovery agents have the privilege to insert, delete, and search for key records. The search feature works like this:

- When the key recovery agents search by the key ID, only the key that corresponds to that ID is returned.
- When the agents search by user name, all stored keys belonging to that owner are returned.
- When the agents search by the public key in a certificate, only the corresponding private key is returned.

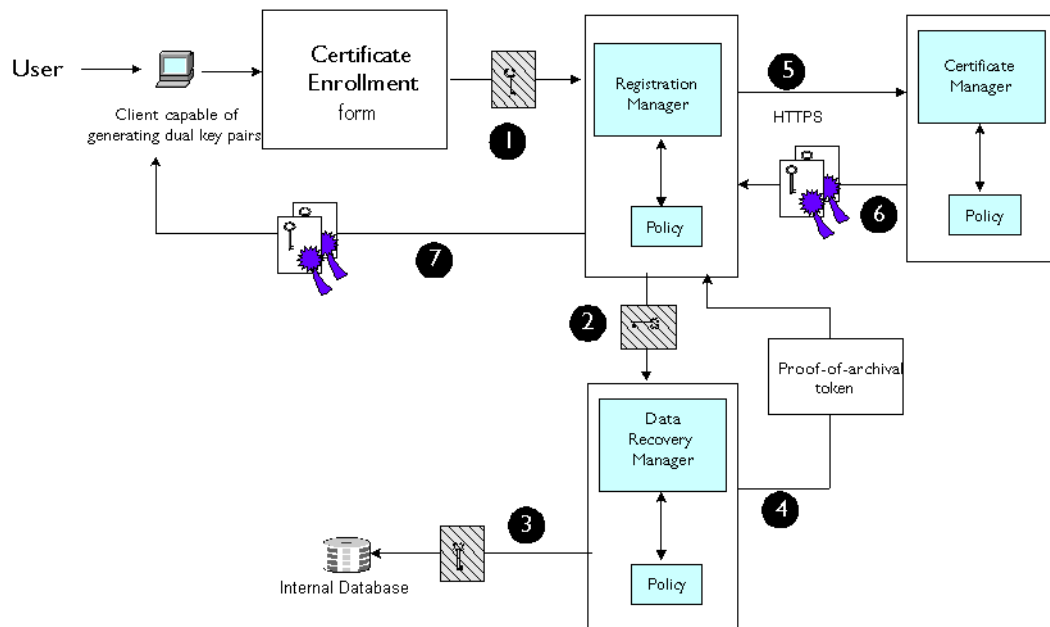
## How Key Archival Works

When a Certificate Manager or Registration Manager receives a certificate request that contains the key archival option, it automatically requests the service of the Data Recovery Manager to archive the user's encryption private key. The Data Recovery Manager receives an encrypted copy of the user's private key and stores the key in its key repository. To archive the key, the Data Recovery Manager uses two special key pairs:

- A transport key pair and corresponding certificate
- A storage key pair

Figure 25.1 illustrates how the key archival process occurs when a user requests a certificate. The deployment scenario shown in this figure has a Registration Manager acting as the trusted enrollment authority to a Certificate Manager and Data Recovery Manager.

Figure 25.1 How the key archival process works



These are the steps shown in Figure 25.1:

1. A user uses a client capable of generating dual key pairs to access the certificate enrollment form served by the Registration Manager, fills in all the information, and submits the request.

The Registration Manager detects the key archival option in the user's request and asks the client for the user's encryption private key.

The client encrypts the user's encryption private key with the public key from the Data Recovery Manager's transport certificate; a copy of the transport certificate is embedded in the enrollment form.

2. Upon receiving the encrypted key from the client, the Registration Manager sends it to the Data Recovery Manager for storage, along with some other information (including the user's public key). Then, the Registration Manager waits for verification from the Data Recovery Manager that the private key has been received and stored and that it corresponds to the user's public encryption key.

3. Upon receiving the encrypted key from the Registration Manager, the Data Recovery Manager decrypts it with the private key that corresponds to the public key in its transport certificate. After confirming that the private encryption key corresponds to the user's public encryption key, the Data Recovery Manager encrypts it again with its storage key before storing it in its internal database. (The storage key either resides in a software or a hardware token and is never exposed to any other entity.)
4. Once the user's private encryption key has been successfully stored, the Data Recovery Manager uses the private key of its transport key pair to sign a token confirming that the key has been successfully stored; the Data Recovery Manager then sends the token to the Registration Manager.
5. After the Registration Manager receives and verifies the signed token, it sends the certificate request to the Certificate Manager for issuance.
6. The Certificate Manager formulates two certificates, one each for signing and encryption key pairs, and returns them to the Registration Manager.
7. The Registration Manager forwards the certificates to the client (the user).

Note that all three subsystems subject the request to configured policy rules at appropriate stages. If the request fails to meet any of the policy rules, the subsystem rejects the request.

## Key Recovery Process

The Data Recovery Manager supports agent-initiated key recovery. In this method of key recovery, designated recovery agents use the Key Recovery form provided in the Data Recovery Manager Agent Services interface to process key recovery requests, list archived keys, and approve recovery. With the approval of a specified number of agents, an organization can recover keys when the key's owner is unavailable or when keys have been lost.

## Key Recovery Agents and Their Passwords

*Key recovery agents* have the authority to retrieve end users' encryption private keys. The recovery agent's role can be performed by any person in your organization. As system administrator, you can designate one or more individuals to be key recovery agents. These individuals need to do the following:

- They must specify a secure password, which in combination with other recovery agents' passwords will be used for protecting the database in which the Data Recovery Manager stores users' keys. You facilitate this by allowing each recovery agent to enter a password in the Data Recovery Manager configuration.
- They must be available to retrieve your users' encryption private keys if the need arises. It isn't necessary for all key recovery agents to be available for the key recovery operation. You specify how many agents are required to authorize the recovery of a key; see "Key Recovery Agent Scheme" on page 636. However, the specified number of key recovery agents must all provide their passwords at the same time to authorize the recovery of a specific key.

The first time you create key recovery agents and specify their passwords is during the installation of the Data Recovery Manager. However, you can change the number of recovery agents and their passwords later by modifying it in the Data Recovery Manager configuration; see "Changing Key Recovery Agents' Passwords" on page 639.

## Secret Sharing of Storage Key Password

The Data Recovery Manager uses the private key of its *storage key pair* to encrypt the repository where it store users' encryption private keys. This requires that the storage key be well protected. For the protection of the storage key pair, the Data Recovery Manager supports a password-splitting mechanism called *m of n secret splitting or sharing*, whereby it splits the PIN that protects the token in which the storage key pair resides among *n* number of key recovery agents and reconstructs the PIN only if *m* number of recovery agents provide their individual passwords; *n* must be an integer greater than 1 and *m* must be an integer less than or equal to *n*.

Here's how the  $m$  of  $n$  secret splitting mechanism gets built and works:

During the installation of a Data Recovery Manager, you generate the storage key pair and specify the hardware token in which the key pair is to be stored. At this time, you also specify a PIN (or password) to protect the token, the total number of key recovery agents ( $n$ ), and how many of these agents ( $m$ ) are required to perform a key recovery operation. You can change the  $m$  of  $n$  secret splitting later; for details, see “Key Recovery Agent Scheme” on page 636.

The Data Recovery Manager splits the PIN for the token into  $n$  parts or pieces. It then encrypts these parts with the passwords that are provided by the authorized key recovery agents.

During the key recovery procedure, the required number of key recovery agents ( $m$ ) provide their identifiers and passwords. After verifying the passwords, the Data Recovery Manager reconstructs the PIN for the token based on the given information.

## Interface for the Key Recovery Process

With the Key Recovery form provided in the Data Recovery Manager Agent Services interface, key recovery agents can collectively unlock the key repository of the Data Recovery Manager and retrieve end users' encryption private keys and associated certificates in a PKCS #12 package, which can then be imported into the client. For an overview of this process, see “How Agent-Initiated Key Recovery Works” on page 633.

Because key recovery agents use the Data Recovery Manager Agent Services interface, agent-initiated key recovery invariably involves the Data Recovery Manager agent and key recovery agents. The Data Recovery Manager agent's certificate is required to access the Key Recovery form, and key recovery agents' passwords are required to unlock the key repository. For information on Data Recovery Manager agents, see “Agents” on page 135.

Your organization's PKI policy may require that the key recovery process be restricted to authorized recovery agents only, preventing any Data Recovery Manager agent from being involved. If so, you should ask all key recovery agents to get client certificates and set them up as Data Recovery Manager agents. For instructions, see “Setting Up Agents” on page 152.

## Local Versus Remote Key Recovery Authorization

Key recovery agents can authorize the recovery of a key locally or remotely. The overview of local and remote authorization provided in this section is intended to help you determine which to use for your organization. You may find it useful to go over the Data Recovery Manager agent-specific information in the *Netscape Certificate Management System Agent's Guide*. An online version of this guide is located here:

```
<server_root>/cert-<instance_id>/web/agent/manual/agt_gide
```

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

`<instance_id>` is the ID for this instance of Certificate Management System. You first specified this during installation.

### Local Key Recovery Authorization

To initiate key recovery locally, the required number of recovery agents assemble in front of the host system that allows them to access the Data Recovery Manager Agent Services interface. Either a Data Recovery Manager agent or a key recovery agent with a Data Recovery Manager agent certificate accesses the Key Recovery form hosted by the Data Recovery Manager and initiates the key recovery process. All key recovery agents enter their IDs and passwords on the same Recovery Authorization form presented by the Data Recovery Manager. If the information presented is correct, the Data Recovery Manager retrieves the requested key and returns it along with the corresponding certificate in the form of a PKCS #12 package.

By default, key recovery authorization is local.

### Remote Key Recovery Authorization

To authorize key recovery remotely, the required number of recovery agents access the Data Recovery Manager Agent Services interface at their own locations and use the Authorize Recovery button to enter each authorization separately.

Before key recovery agents can authorize key recovery remotely, they must be set up to function as Data Recovery Manager agents. This role gives them the privilege to access the Data Recovery Manager's Agent Services interface directly.



In remote key recovery authorization, one of the key recovery agents informs all required recovery agents about an impending remote key recovery process. All recovery agents access the Key Recovery page hosted by the Data Recovery Manager. One of the agents initiates the key recovery process. The Data Recovery Manager returns a notification to each agent. The notification includes a recovery authorization reference number identifying the particular key recovery request that the agent is required to authorize. Each agent uses the reference number and authorizes key recovery separately.

The Data Recovery Manager informs the agent who initiated the key recovery process of the status of the authorizations. When all of the authorizations are entered, the Data Recovery Manager checks the information. If the information presented is correct, it retrieves the requested key and returns it along with the corresponding certificate in the form of a PKCS #12 package to the agent who initiated the key recovery process.

Key recovery agents can switch to remote authorization by deselecting the local authorization option in the Key Recovery form.

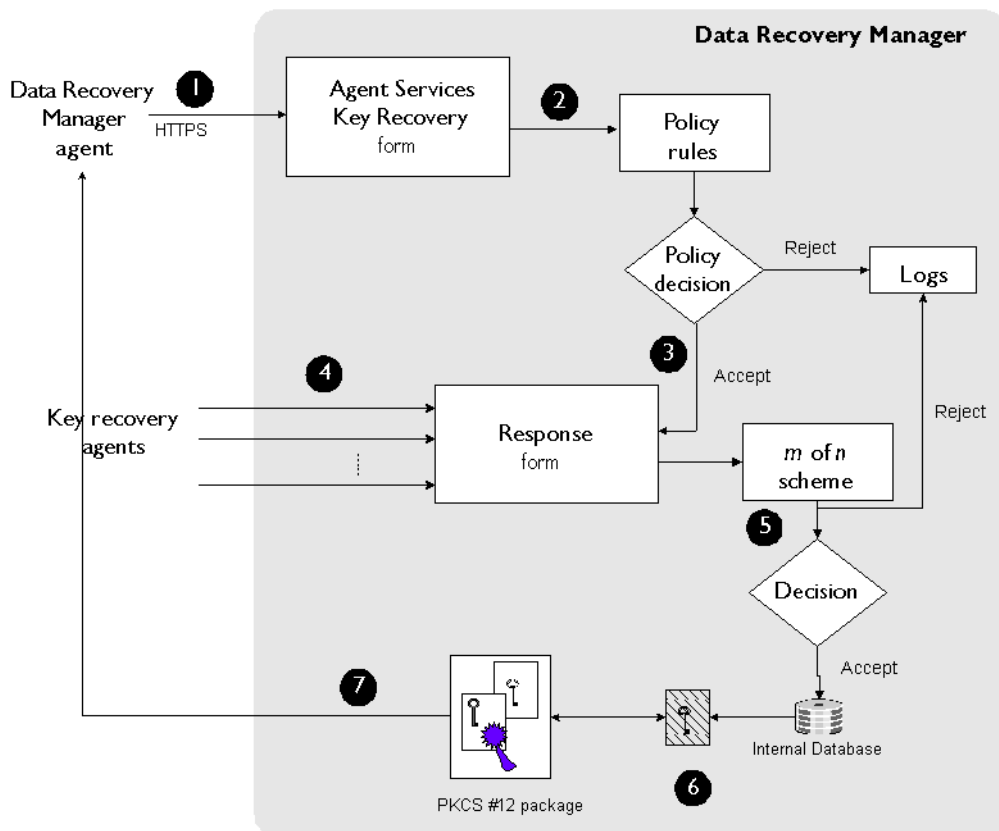
## How Agent-Initiated Key Recovery Works

In an agent-initiated key recovery, the key is recovered by the collective efforts of a Data Recovery Manager agent and authorized key recovery agents. You may need to resort to this type of key recovery if the owner of a key cannot be reached and the authorities in the organization need to access that user's encrypted data (for example, S/MIME mail messages).

Upon retrieving the private encryption key (in the form of a PKCS #12 package), the agents may forward the key to the original user, the manager of the original owner, or some other authorities.

Figure 25.2 illustrates how agent-initiated key recovery works.

Figure 25.2 The agent-initiated key recovery process



These are the steps shown in Figure 25.2:

1. The Data Recovery Manager agent accesses the Key Recovery form using the appropriate client certificate, types the identification information pertaining to the person whose encryption private key needs to be recovered, and submits the request.

The request is submitted to the Data Recovery Manager over HTTPS.

2. The Data Recovery Manager subjects the key recovery request to its policy checks.

3. If the request passes all the policy rules, the Data Recovery Manager sends a confirmation HTML page to the web browser the agent used. If the request fails any of the policy checks, the server logs an appropriate error message.

The confirmation page contains information and input sections:

- The information section includes the user's information.
- The input section includes fields for entering the user's certificate corresponding to the key that needs to be recovered, the password for the PKCS #12 package, and key recovery agents' passwords.

The Data Recovery Manager uses the certificate to construct the PKCS #12 package (which includes the user's encryption private key and corresponding certificate), the PKCS #12 password to encrypt the PKCS #12 package, and key recovery agents' passwords to construct the PIN required to unlock its key repository.

4. The key recovery agents verify the information in the confirmation page and enter the certificate in MIME-64 format, the password for the PKCS #12 package, and their individual identifiers and passwords. The Data Recovery Manager agent submits the page to the Data Recovery Manager.
5. The Data Recovery Manager matches the key recovery agent information with its *m of n scheme* (see "Key Recovery Agent Scheme" on page 636). After verifying that the required number of recovery agents entered their passwords, the server uses the agents' passwords to construct the PIN required to access the private key repository.
6. The Data Recovery Manager then retrieves the user's private key from its key repository and decrypts it by using the private component of the storage key pair.
7. The Data Recovery Manager packages the user's certificate and the corresponding private key as a PKCS #12 package and encrypts it with the PKCS #12 password provided by the recovery agent. It then delivers the package to the client the recovery agent used to initiate the key recovery process, and prompts the agent to store the encrypted package. The agent may choose to store the package in the local file system of the client machine (only if it has restricted access) or on a floppy diskette.

The recovery agent can then send the encrypted PKCS #12 package and the corresponding password to an individual by any secure, out-of-band means.

**Important** The PKCS #12 package contains the private key. To minimize the risk of key compromise, the recovery agent must use any secure, out-of-band means to deliver the PKCS #12 package and password to the key recipient. As an administrator, you should recommend the recovery agent to use a good password for encrypting the PKCS #12 package, and also consider setting up an appropriate delivery mechanism.

## Key Recovery Agent Scheme

The *key recovery agent scheme* consists of configuring the Data Recovery Manager to recognize a fixed number of key recovery agents (a minimum of one) and specifying how many of these agents are required to authorize a key recovery request before the archived key is restored. Each recovery agent provides the Data Recovery Manager with a password, which it uses to generate a unique PIN; the Data Recovery Manager uses the PIN to protect its storage key pair, which in turn protects users' keys.

The Data Recovery Manager tracks the key recovery agent password for each agent and allows you to facilitate changing agents' passwords; you do not have direct access to these passwords or the actual storage key password. Each password retrieves only a part of the private storage key.

You first specified the key recovery agent scheme when you installed the Data Recovery Manager.

## Changing the Key Recovery Agent Scheme

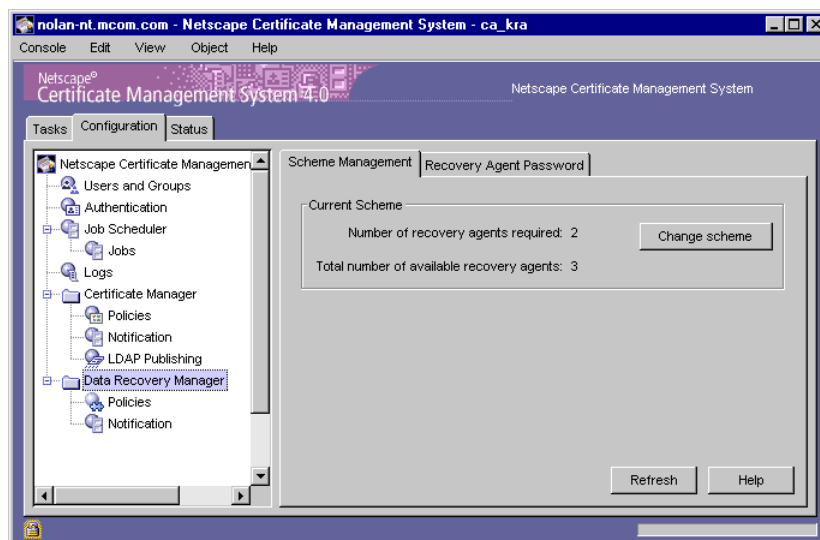
You can change the total number of key recovery agents for a Data Recovery Manager and the number of key recovery agents required to retrieve an end user's encryption private key from the Data Recovery Manager's key repository.

To change the key recovery agent scheme:

1. Access the CMS window (see "Accessing the CMS Window" on page 63).
2. Click the Configuration tab.

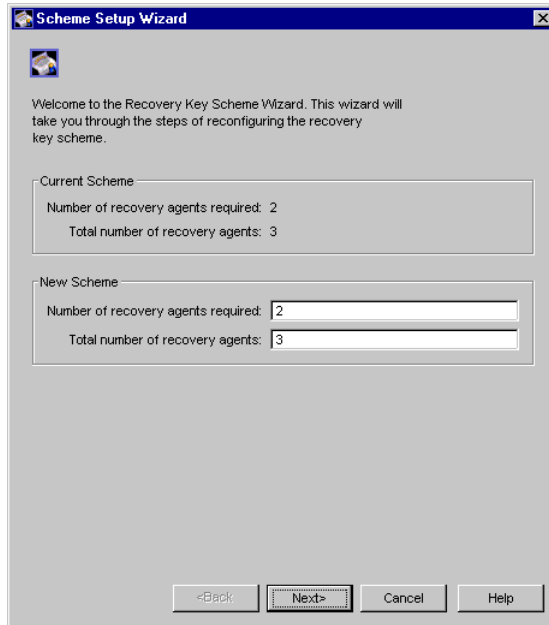
3. In the navigation tree, select the Data Recovery Manager, and in the right pane, click the Scheme Management tab.

The Scheme Management tab shows the current key recovery scheme.



4. Click Change scheme.

The Change Recovery Key Scheme window appears.



The image shows a Windows-style dialog box titled "Scheme Setup Wizard". It has a blue title bar with a close button. Inside, there is a small icon of a wizard. The text reads: "Welcome to the Recovery Key Scheme Wizard. This wizard will take you through the steps of reconfiguring the recovery key scheme." Below this, there are two sections: "Current Scheme" and "New Scheme". The "Current Scheme" section shows "Number of recovery agents required: 2" and "Total number of recovery agents: 3". The "New Scheme" section has two input fields: "Number of recovery agents required:" with the value "2" and "Total number of recovery agents:" with the value "3". At the bottom, there are four buttons: "<Back", "Next>", "Cancel", and "Help".

5. In the New Scheme section, make the appropriate changes:

**Number of recovery agents required.** Type the number of agents required to authorize a key recovery process. The number cannot be zero and must be equal to or less than the total number of recovery agents.

**Total number of recovery agents.** Specify the total number of key recovery agents. The number cannot be less than one and must be equal to or greater than the number of agents required to authorize the key recovery operation.

6. Click Next.
7. For each agent, enter a user name and password, then click Next.

The number of screens depends on the total number of agents you have specified.

8. When you have entered all agent information, click Finish.

You are returned to the Scheme Management tab.

## Changing Key Recovery Agents' Passwords

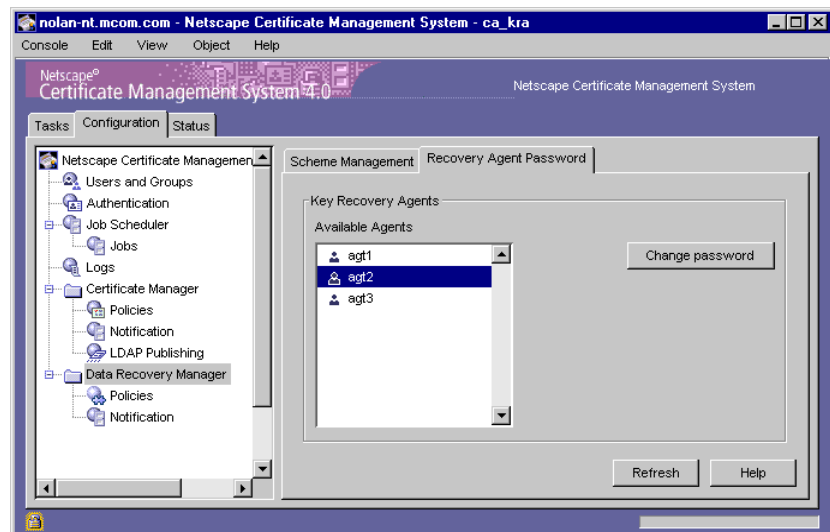
As administrator, you have the responsibility of safeguarding the security of each Data Recovery Manager installed in your PKI setup. One of the safety measures you can implement is to ask your key recovery agents to change their passwords periodically. This way, you will be sure that the required number of agents are available if a key needs to be recovered. If key recovery agents routinely change their passwords, they are less likely to forget them.

The CMS window allows you to view the list of currently designated key recover agents and, if necessary, change an agent's password.

To change key recovery agents' passwords:

1. Access the CMS window (see “Accessing the CMS Window” on page 63).
2. Click the Configuration tab.
3. In the navigation tree, select the Data Recovery Manager, and in the right pane, click the Recovery Agent Password tab.

The tab shows current key recovery agents in the Available Agents list.



4. Select the agent whose password needs to be changed, and click Change Password.

The Change Password dialog box appears.



5. Allow the agent to enter the appropriate information.

During installation, the Data Recovery Manager prompts you to enter key recovery agent passwords. The number of passwords you must enter depends on the key recovery agent scheme you chose; for details, see “Key Recovery Agent Scheme” on page 636. If you are changing a password for the first time after installation, in the Old password field you must enter the recovery agent password you specified during installation. Then in the remaining fields, allow the key recovery agent to enter the new password information. If you have more than one key recovery agent, repeat this procedure for all the agents.

**Old Password.** Type the current password for the key repository.

**New Password.** Type the new password for the key repository.

**Confirm Password.** Retype the new password exactly as you typed it in the previous field.

6. Click OK.

You are returned to the Recovery Agent Password tab.



# Setting Up Key Archival and Recovery Process

By default, the Data Recovery Manager is not configured to archive or recover end users' encryption private keys. This section explains how to set up key archival and recovery processes.

- Setting Up the Key Archival Process
- Setting Up the Key Recovery Process

## Setting Up the Key Archival Process

Before proceeding with this section, you should have read “Key Archival Process” on page 626. In particular, you should be familiar with how the key archival process works. If you are not, see “How Key Archival Works” on page 627.

Setting up key archival involves the following steps:

- Step 1. Deploy Clients That Can Generate Dual Key Pairs
- Step 2. Connect the Enrollment Authority and the Data Recovery Manager
- Step 3. Customize the Certificate Enrollment Form
- Step 4. Configure Key Archival Policies
- Step 5. Test Your Key Archival Setup

### Step 1. Deploy Clients That Can Generate Dual Key Pairs

You can use the Data Recovery Manager to archive and recover keys only from clients that support dual key-pair generation, the key archival option, and the CMC protocol. Clients that do not meet this criteria cannot be used with the Data Recovery Manager. To understand why you need to use clients that can generate dual key pairs, see “Clients That Can Generate Dual Key Pairs” on page 624.

The domestic version of Certificate Management System includes a plug-in called *Dual-Key Test Bed*, which when plugged into Netscape Communicator version 4.5 enables it to support the CMC protocol and generate dual key pairs. For information on installing the plug-in, see the *Dual-Key Test Bed Release Notes*. It is at this location:

```
<server_root>\bin\cert\nsm\NSM_RELNOTES.html
```

`<server_root>` is the directory where the Data Recovery Manager's binaries are kept. You first specified this directory during installation.

For the most current information on *Dual-Key Test Bed*, check this site:

```
http://home.netscape.com/eng/server/cms
```

## Step 2. Connect the Enrollment Authority and the Data Recovery Manager

Key archival occurs when dual key pairs are generated by the client. The client generates the key pairs when a user requests a certificate by filling out the appropriate certificate enrollment form served by an enrollment authority, which can be either a Certificate Manager or a Registration Manager. When the enrollment authority detects the key archival option in the request, it initiates the key archival process and requests the service of the Data Recovery Manager for archiving the key.

For the enrollment authority to be able to request the service of the Data Recovery Manager, the two subsystems must be configured to recognize, trust, and communicate with each other. When you installed the Data Recovery Manager, you were asked to connect it to a Certificate Manager or Registration Manager. You might have specified some of the configuration information required for the two subsystems to communicate with each other. Also, if the enrollment authority and the Data Recovery Manager are installed in the same CMS instance, certain configurations are done automatically.

However, to ensure that key archival takes place successfully, you must make sure that the Data Recovery Manager is connected to the appropriate enrollment authority. Also verify whether the enrollment authority has been set up as a privileged user, with an appropriate SSL client authentication certificate, in the internal database of the Data Recovery Manager. By default, the Certificate Manager uses its *SSL server certificate* for SSL client authentication,

whereas the Registration Manager uses its *signing certificate* for this purpose; for more information, see “Keys and Certificates for the Main Subsystems” on page 184.

Otherwise, follow the instructions in “Setting Up Trusted Managers” on page 158 and set up the enrollment authority as a trusted front end to the Data Recovery Manager.

### Step 3. Customize the Certificate Enrollment Form

For the enrollment authority to automatically initiate the key archival process at the time key pairs are generated, a certificate request must include the following information:

- The key archival option—this must be included in the certificate enrollment form that your users use to request certificates.
- The Data Recovery Manager’s transport certificate—this must also be included in the certificate enrollment form. The Data Recovery Manager uses it to encrypt the user’s encryption private key with the public key in the transport certificate before sending the user’s key to its key repository. For information about the key repository, see “Where the Keys are Stored” on page 626.

Make sure that the transport certificate, in base-64 encoded format, is embedded in the form. Otherwise, the Data Recovery Manager will fail to archive users’ keys.

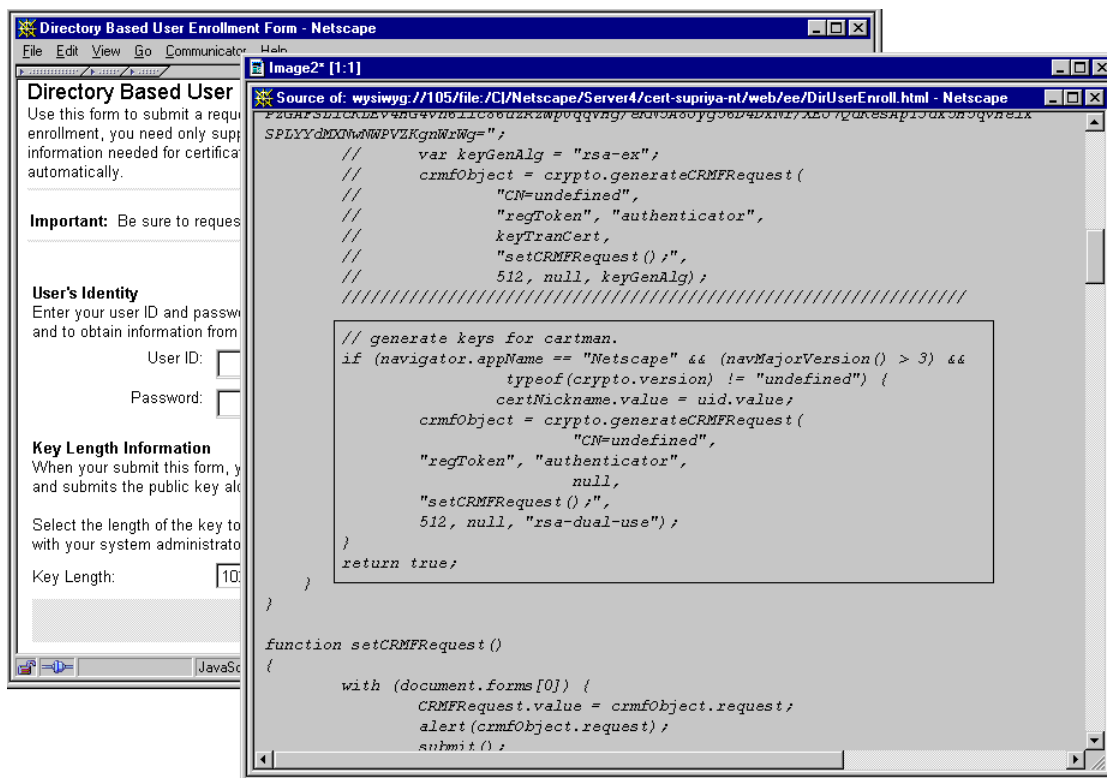
All the three end user enrollment forms provided by Certificate Management System—the directory-based enrollment form (`DirUserEnroll.html`), directory- and PIN-based enrollment form (`DirPinUserEnroll.html`), and manual enrollment form (`ManUserEnroll.html`)—contain the necessary JavaScript code for initiating the key archival process; for details about these forms, see “Forms for Certificate Enrollment” on page 555. If you are using any of these forms for end-user enrollment, make sure to update the `generateCRMRequest()` JavaScript method. If you plan to use custom enrollment forms for users, be sure to include the required JavaScript code in those forms.

Figure 25.3 shows the default directory-based enrollment form with the information related to the `generateCRMRequest()` JavaScript method highlighted. Note that the JavaScript method includes parameters for specifying various things. You are required to update the following information only:

- The Data Recovery Manager's transport certificate
- The algorithm, length, type, and usage for the end user key pair. When you update this information, the key archival option is automatically set.

The steps that follow explain how to do this.

Figure 25.3 Data Recovery Manager's transport certificate information in the default enrollment form



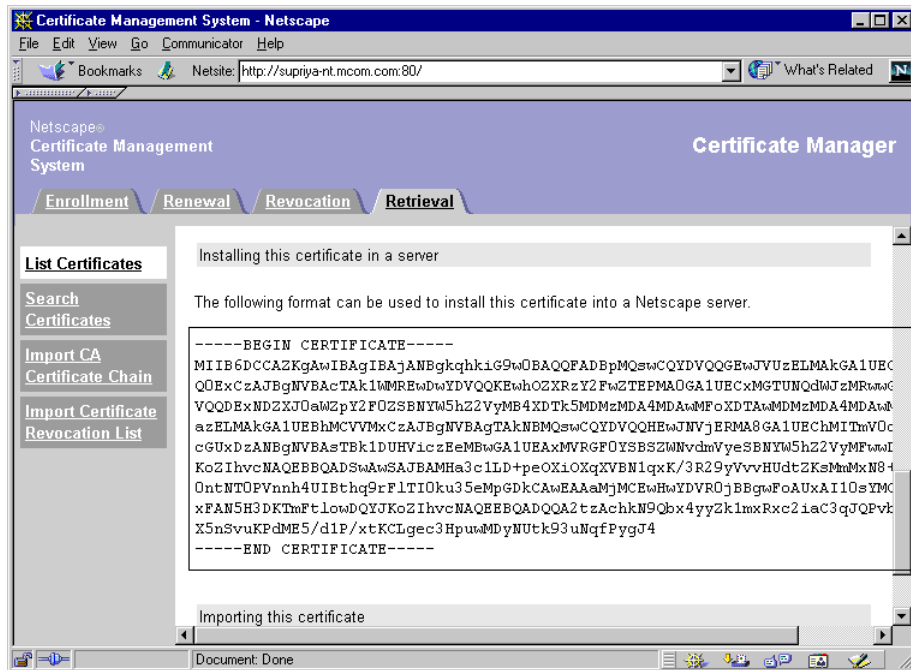
## **I. Copy the transport certificate in its base-64 encoded format**

The transport certificate is stored in the Data Recovery Manager's certificate database. If the transport certificate is signed by a Certificate Manager, then a copy of the certificate is also available with the Certificate Manager. Follow the instructions as appropriate.

- To copy the transport certificate information from a Certificate Manager's database:
  1. Open a web browser window.
  2. Go to the end-entity page hosted by the Certificate Manager.
  3. Click the Retrieval tab.
  4. List or search for the transport certificate.
  5. Click Details, and view the certificate information.

Make sure that the certificate you are looking at is the correct one; the certificate shows the DN that was specified for the transport certificate during the installation of Data Recovery Manager.

6. Scroll down to the section that says “Installing this certificate in a server.”



- Copy the base-64 encoded certificate, *excluding* the marker lines -----BEGIN CERTIFICATE----- and -----END CERTIFICATE-----, to a text file. An example is shown below:

MI ICDjCCAXegAwIBAgIcAFmWdQYJKoZIhvcNAQEEBQAwZDZELMAKGAlUEBHMCMVVMxLDAQBgNVBA  
oTIO51dhmJjYXB1IENvbWw1bmbl jYXRpb25z IENvcnBvcF0AOW9uMRERWdYVQ0XDEtYXJkY29y  
ZTEhENUCGAlUEBHMSEGFYzGNvcmUgQ2YydgGAlmWnhdGUGU2Y2dmYyVlElJmB4QDk7t4MTEoOTiZnd  
IXOvOXDtk5MDUXODiZNDiIvOXvowLjELMAKGAlUEBHMCMVVMxLDAQEPABBgNVBAoTCG51dhmJjYXB1MQww  
CgYDVQDQwEwNlUmEwZDANBgkqhkiG9w0BAQEFAANLADBiAkeAarrbdiYU1S6dLCKKAoBEBn1m83  
k63b6dhytRYNkHDB5Bp85SRadmdJV+00YmXjYAU1GCFrmccY42sh2YSvowIscDQABozYWNDRABg  
lgkhgBhvhCAQEEBAMCAAwHwYDVRO jBgGwFAU17ftrscYCF1MQ19f jmm3LnNu3oAwDQYJKoZIh  
vcNAQEEBQADgYEApvzcUsVIOv4oYSiWb4+aMVH6s1jiJLr5iVhnoKzfSxYxPvDUw6uz04AT8N+  
KIaRMtKXHPDzGAFSLicLkLev4HG4v6h1l086uzRzWpUqVHgeKN5A8Jyg56D4DkNRXej7QdKesa  
p13dK55vH6fALSPLYdMXNWNP

- To copy the transport certificate information from a Data Recovery Manager's certificate database:
  1. Open a terminal window in the system that hosts the Data Recovery Manager.
  2. Use the command-line tool called `certutil` to retrieve the transport certificate from the Data Recovery Manager's certificate database. (For information on the `certutil` tool, see Appendix D, "Certificate Database Tool" in the online version of this document.)

First, go to `<server_root>/cert-<instance_id>/config`

Next, run `<server_root>/bin/cert/tools/certutil -L -d . -n kraTransportCert <instance_id> -a`

`<server_root>` is the directory where the Data Recovery Manager's binaries are kept. You first specified this directory during installation.

`<instance_id>` is the ID for this instance of the Data Recovery Manager. You first specified this when you installed this server.

The transport certificate appears. View the certificate information. Make sure that the certificate you are looking at is the correct one; the certificate shows the DN that was specified for the transport certificate during the installation of Data Recovery Manager.

3. Copy the base-64 encoded certificate, *excluding* the marker lines `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----`, to a text file. An example is shown below:

```
MIICDjCCAXegAwIBAgICAfMwDQYJKoZIhvcNAQEEBQAwZDZELMAkGA1UEBhMCVVMxLDAqBgNVBA
oTIO5ldHNjYXB1IENvbW11bmljYXRpb25zIENvcnBvcnF0aW9uMREwDwYDVQQLEWhlYXJkY29y
ZTEuMCUGA1UEAxMeSGFyZGNvcnUgQ2VydGhmaWNhdGUU2VydmVzIElJMB4XDtk4MTEzNDIzND
IxOVoXDTk5MDUxODIzNDIxOVowLjELMAkGA1UEBhMCVVMxETAPBgNVBAoTCG5ldHNjYXB1MQww
CgYDVQQDEWNLUmEwXDANBgkqhkiG9w0BAQEFAANLADBIAkEArrbdiYUI5SCd1CKKa0bEBn1m83
kX6bdhytRYNkdHB95Bp85SRadmdJV+0OyMxjYAtGCFrmcqEZ4sh2YSov6wIDAQABozYwNDARBg
lghkgBhvCAQEEBAMCAEAWHwYDVR0jBBgwFoAU17FtsrYCF1QM19fjMm3LnNu3oAwDQYJKoZIh
vcNAQEEBQADgYEApvzcUsVIOstaoYSiWb4+aMVH6s1jiJ1r5iVHnOKzfsYxPVdUw6uz04AT8N+
1KIarMTKxHPzGAFSLicKLEv4HG4vh61lc86uzRzWpUqqqVHgeKN5A8Jyg56D4DkNrXEJ7QdKesa
p13dk5H5qvHelkSPLYdMXNwNWP
```

## 2. Update the JavaScript method in the enrollment form

1. Go to the host system of the enrollment authority and locate the user-enrollment form. The default forms are at this location:

```
<server_root>/cert-<instance_id>/web/ee/...
```

<server\_root> is the directory where the enrollment authority's binaries are kept. You first specified this directory during installation.

<instance\_id> is the ID for this instance of the enrollment authority. You first specified this when you installed this server.

2. Open the enrollment form in an HTML or text editor.
3. In the form, locate the `generateCRMRequest()` JavaScript method (see Figure 25.3).
4. Add a variable for the transport certificate.

Below the commented text, add this line:

```
var kraTransportCert =
```

5. Open the text file that has the Data Recovery Manager's transport certificate (the one you copied earlier) and copy the certificate.
6. Paste the certificate as the value of the `kraTransportCert` variable.

Paste the certificate in front of the `=` sign, remove any line breaks, enclose the certificate within double-quotation marks, and end the string with a semicolon (`;`). When deleting line breaks, be sure not to delete any of the characters in the encoded blob.

An example is shown below:

```
var kraTransportCert =
"MIICDjCCAXegAwIBAgICAfMwDQYJKoZIhvcNAQEEBQAwdzELMAkGA1UEBhMCVVMxLDAqBgNVB
AoTIO5ldHNjYXB1IENvbWl1bm1jYXRpb25zIENvcnBvcnF0aW9uMRERDwYDVQLEWhIYXJkY29
yZTENMCUGA1UEAxMeSGFyZG9uU2VydG1maWNhdGU2V2YdmVYIElJMB4XDTk4MTEExOTIzN
DIxOVoXDTk5MDUxODIzNDIxOVowLjELMAkGA1UEBhMCVVMxETAPBgNVBAoTCG5ldHNjYXB1MQw
wCgYDVQQDEWNLUmEwXDANBgkqhkiG9w0BAQEFAANLADBIaKEArrbDiYUI5SCdlCKKa0bEBn1m8
3kX6bdhytRYNkdHB95Bp85SRadmdJV+00yMxjYAtGCFrmcqEz4sh2YSov6wIDAQABozYwNDARB
glghkgBhvhCAQEEBAMCAEAWHwYDVR0jBBgwFoAU17FtsrYCF1QM19fjMm3LnNu3oAwDQYJKoZI
hvcNAQEEBQADgYEApvzcUsVIOstaoYSiWb4+aMVH6s1jiJlr5iVhNOKzfsYxPVdUw6uz04AT8N
+1KIarMTKxHPzGAFSLicKLEv4HG4vh6llc86uzRzWpUqqVHGekN5A8Jyg56D4DkNrXEJ7QdKes
Ap13dk5H5qvHelkSPLYdMXNwNWP";
```



7. Pass the `kraTransportCert` variable to the JavaScript method.

Replace `null` (the fourth line in the method) with `kraTransportCert`.

8. Specify the key algorithm and key type.

For information on specifying key type, length, and algorithm, see “generateCRMFRequest()” in *Javascript API for Client Certificate Management*. The document is at this location:

```
<server_root>/bin/cert/nsm/CMCJavascriptAPI.html
```

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

Below is an example that shows how the updated `generateCRMFRequest()` method would look:

```
// generate keys for nsm.

if (navigator.appName == "Netscape" && (navMajorVersion() >
    3) &&
    typeof(crypto.version) != "undefined") {
    certNickname.value = subject.value;

    crmfObject = crypto.generateCRMFRequest(subject.value,
        "regToken",
        "authenticator",
        kraTransportCert,
        "setCRMFRequest()";",
        512, null, "rsa-ex",
        1024, null, "rsa-sign");
}
```

The method triggers the client to generate two RSA key pairs—one key of length 512 for encrypting data and another key of length 1024 for signing data.

9. Save your changes.

## Step 4. Configure Key Archival Policies

This step is optional.

Unlike Certificate Manager and Registration Manager, no policy plug-in modules are provided for the Data Recovery Manager. If you have implemented any custom policy modules for the Data Recovery Manager's key archival process, you should make sure that they are configured properly. For details on configuring policies for a subsystem, see "Configuring Policies" on page 459.

## Step 5. Test Your Key Archival Setup

To test whether you can successfully archive a key:

### 1. Enroll for dual certificates

1. Open a web browser window using the client you deployed in Step 1.
2. Go to the end-entity interface for the enrollment authority. The default URL is as follows:

```
https://<host_name>:<end_entity_HTTPS_port> or  
https://<host_name>:<end_entity_HTTP_port>
```

3. Open the enrollment form you customized in Step 3.
4. Fill in all the values and submit the request.  
The client prompts you to enter the password for your key database.
5. When you enter the correct password, the client generates the key pairs.  
Do not interrupt the key-generation process.

### 2. Approve the request

This step is required only if you used the manual enrollment form for requesting the certificates.

1. Go to the enrollment authority's Agent Services interface.

The default URL is as follows:

```
https://<host_name>:<agent_port>
```

2. Click the link that says List Requests.
3. In the form that appears, select the “Show pending requests” option and click Find.

You should see your request in the list of pending requests.

4. Locate and approve the request.

### **3. Check if the certificates have been issued**

1. Click the List Requests link again.
2. In the form that appears, select the “Show completed requests” option and click Find.

You should see two new certificates with consecutive serial numbers.

### **4. Check if the key has been archived**

1. Go to the Data Recovery Manager’s Agent Services interface.
2. Click the link that says List Requests.
3. In the form that appears, check the “Show completed requests” option and click Find.
4. If the key has been archived successfully, you should see the information pertaining to that key.

## **Setting Up the Key Recovery Process**

Before proceeding with this section, you should have read “Key Recovery Process” on page 629. In particular, you should be familiar with how the key archival process works. If you are not, see “How Agent-Initiated Key Recovery Works” on page 633.

The Data Recovery Manager supports agent-initiated key recovery process, in which end users’ encryption private keys are recovered by designated key recovery agents. This section explains how to set up the key recovery process.

Setting up agent-initiated key recovery involves the following steps:

- Step 1. Verify the  $m$  of  $n$  scheme
- Step 2. Facilitate the Key Recovery Agents to Change the Passwords
- Step 3. Determine the Authorization Mode for Key Recovery
- Step 4. Customize the Key Recovery Form
- Step 5. Configure Key Recovery Policies
- Step 6. Test Your Key Recovery Setup

## Step 1. Verify the $m$ of $n$ scheme

During the installation of the Data Recovery Manager, you were asked to specify the total number of key recovery agents (a minimum of one) and the number of agents (of this total) required to authorize a key recovery operation. This combination is called  $m$  of  $n$  scheme. For more information about this, see “Key Recovery Agent Scheme” on page 636.

Verify that the current  $m$  of  $n$  scheme is appropriate for your PKI setup. If it isn't, change the scheme following the instructions in “Changing the Key Recovery Agent Scheme” on page 636.

## Step 2. Facilitate the Key Recovery Agents to Change the Passwords

During the installation of Data Recovery Manager, after you specified the  $m$  of  $n$  scheme, you were also prompted to provide unique passwords for each recovery agent. It is quite likely that you specified these passwords yourself instead of it being done by those individuals who have been designated with the key recovery agents' role in your organization. Therefore, you must get the designated recovery agents to change the passwords entered during installation.

- To understand the significance of key recovery agents' passwords, see “Key Recovery Agents and Their Passwords” on page 630.
- To get the recovery agents to change the passwords, follow the instructions in “Changing Key Recovery Agents' Passwords” on page 639.

### Step 3. Determine the Authorization Mode for Key Recovery

The Data Recovery Manager allows key recovery agents to authorize recovery of an end user's encryption private key locally or remotely. The default configuration is local authorization. It is important that you evaluate both the authorization modes, and choose the one that is appropriate for your organization. For more information about this, see "Local Versus Remote Key Recovery Authorization" on page 632.

If you want the key recovery agents to authorize key recovery remotely, be sure to set them up as Data Recovery Manager agents following the instructions in "Setting Up Agents" on page 152.

### Step 4. Customize the Key Recovery Form

Key recovery agents need an appropriate interface to initiate the key recovery process. By default, the Data Recovery Manager's Agent Services interface includes an HTML form (`recoverKey.html`) that allows key recovery agents to initiate the key recovery process and retrieve users' encryption keys; for details about this form, see "Summary of Agent Forms and Templates" on page 563.

If you want to customize this form to suit your organization, be careful not to delete any of the information that is vital to the functioning of the form; it is recommended that you restrict your changes to the content presented in the form.

### Step 5. Configure Key Recovery Policies

This step is optional.

Unlike Certificate Manager and Registration Manager, no policy plug-in modules are provided for the Data Recovery Manager. If you have implemented any custom policies for the Data Recovery Manager's key recovery process, you should make sure that they are configured properly. For details on configuring policies for a subsystem, see "Configuring Policies" on page 459.

## Step 6. Test Your Key Recovery Setup

To test whether you can successfully recover an archived key:

1. Open a web browser window.
2. Go to the Data Recovery Manager's Agent Services interface.
3. Click the Recover Keys link.
4. In the form that appears, enter any of the following information for the encryption private key that has been archived:
  - The key owner's name
  - The serial number of the key
  - The public key that corresponds to the private key (in the form of base-64 encoded certificate)
  - The instance ID of the enrollment authority that initiated the key archival process

If you need more information about any of the fields in this form, click the Help button.

5. Click Show Key.

If the key has been archived successfully, you should see the information pertaining to that key.

6. Click Recover.

7. In the form that appears, enter the following information:
  - The PKCS #12 password; the Data Recovery Manager uses this password to encrypt the PKCS #12 package (see “How Agent-Initiated Key Recovery Works” on page 633).
  - The base-64 encoded certificate that corresponds to the private key you want to recover; use the enrollment authority’s end-entity or agent interface to get this information. If you searched for the archived key by providing the base-64 encoded certificate in (step 4), then you don’t have to provide this information.
  - The key recovery agents’ passwords.
8. Click Recover.

If you entered the correct information, the Data Recovery Manager returns the private key packaged as a PKCS #12 blob (it contains the recovered key pair and the corresponding certificate) and prompts you to save it. Specify the path and filename for saving the encrypted file.





# *Appendixes*

Distinguished Names

Backing Up and Restoring Data

Command-Line Utilities

Certificate Database Tool

Key Database Tool

Netscape Signing Tool

SSL Strength Tool

SSL Debugging Tool





# Distinguished Names

This appendix explains what a distinguished name is and how Netscape Certificate Management System (CMS) uses distinguished names to automatically update certificate information in your corporate LDAP directory.

The appendix has the following sections:

- What Is a Distinguished Name? (page 659)
- Role of Distinguished Names in Certificates (page 662)

For the most part, the information presented in this appendix is specific to Netscape Directory Server, an LDAP-compliant directory.

## What Is a Distinguished Name?

Distinguished names (DNs) are string representations that uniquely identify users, systems, and organizations. In general, DN's are used in LDAP-compliant directories, such as Netscape Directory Server. In Certificate Management System, you use DN's to identify the owner of a certificate and the authority that issued a certificate.

**Note** If you are using an LDAP directory in conjunction with Certificate Management System, the DN's in your certificates should match the DN's in your directory.

# Distinguished Name Components

A DN identifies an entry in an LDAP directory. Because directories are hierarchical, DNs identify the entry by its location as a path in a *hierarchical tree* (much as a path in a file system identifies a file). Generally, a DN begins with a specific common name, and proceeds with increasingly broader areas of identification until the country name is specified. DNs are typically made up of the following components (which are defined in the X.520 standard):

CN=common name, [OU=organizational unit, O=organization, L=locality, ST=state or province], C=country name

These components are described in Table 25.1.

Table 25.1 Definitions of standard DN components

Component	Name	Definition
CN	Common name	A required component that identifies the person or object defined by the entry. For example: <ul style="list-style-type: none"><li>CN=Jane Doe</li><li>CN=myhost.mydomain.dom</li></ul>
E (deprecated)	Email address	Identifies the email address of the entry. The use of this component is discouraged.
OU	Organizational unit	Identifies a unit within the organization. For example: <ul style="list-style-type: none"><li>OU=Sales</li><li>OU=Manufacturing</li></ul>
O	Organization	Identifies the organization in which the entry resides. For example: <ul style="list-style-type: none"><li>O=Netscape Communications Corporation</li><li>O=Public Power &amp; Gas</li></ul>

Table 25.1 Definitions of standard DN components (Continued)

Component	Name	Definition
L	Locality	Identifies the place where the entry resides. The locality can be a city, county, township, or other geographic region. For example: <ul style="list-style-type: none"> <li>• L=Mountain View</li> <li>• L=Pacific Northwest</li> <li>• L=Anoka County</li> </ul>
ST	State or province name	Identifies the state or province in which the entry resides. For example: <ul style="list-style-type: none"> <li>• ST=California</li> <li>• ST=British Columbia</li> </ul>
C	Country	Identifies the name of the country under which the entry resides. For example: <ul style="list-style-type: none"> <li>• C=US</li> <li>• C=GB</li> </ul>

**Important** If used in conjunction with an LDAP-compliant directory, Certificate Management System does not recognize components that are not listed here.

## Root Distinguished Name

The root distinguished name, or *root DN*, is the first, or top-most, entry in an LDAP directory tree. In Netscape Directory Server, the root DN is commonly referred to as the *directory manager*. By default, the root DN uses no suffix; it is simply a common name attribute-data pair: `CN=Directory Manager`. For example, the root entry's DN could look like this: `CN=Directory Manager, O=Netscape Communication Corporation, C=US. §`

## Base Distinguished Name

The base distinguished name, or *base DN*, identifies the entry in the directory from which searches initiated by LDAP clients occur; the base DN is often referred to as the search base. For example, if you specify a base DN of

OU=people, O=airius.com for a client, the LDAP search operation initiated by the client examines only the OU=people subtree in the O=airius.com directory tree.

Typically, an LDAP search consists of the following components:

- The base DN—for example, O=Netscape, C=US, which initiates a subtree search through all entries below this entry in the directory (in other words, all entries with the suffix O=Netscape, C=US).
- The search type, which can be a base search (only the entry specified by the base DN is searched), a one-level search (only entries one level below the base entry are searched), or a subtree search (all entries at all levels below the base entry are searched).
- The search filter, which specifies the search criteria applied to each entry within the scope of the search.

When Certificate Management System is configured for LDAP publishing, the search point and search criteria are determined by the configuration parameter values; for details, see information about the mapper or publisher classes in “Directory Update Process” on page 490. In the absence of a base DN value, Certificate Management System uses DN components in the certificate’s subject name to construct the base DN so that it can search the directory in order to publish to or update the appropriate directory entry.

Typically, when you configure Certificate Management System for LDAP publishing, you set the base DN value to `Directory Manager`, so that it can use the publishing directory’s root entry to start searching. This way, the entire directory tree is searched.

## Role of Distinguished Names in Certificates

In certificates issued by Certificate Management System, DN’s are used to identify the entity that owns the certificate. In all cases, if you are using Certificate Management System with a directory, the format of the DN’s in your certificates should match the format of the DN’s in your directory. It is not necessary that the names match exactly; certificate mapping allows the subject DN in a certificate to be different from the one in the directory. For more information, see “Object-Mapping Rules” on page 491.

## DNs in End-Entity Certificates

In end-entity certificates issued by Certificate Management System, DN is used to identify the end entity that owns the certified key pair. The end entity is one of the following:

- The individual who owns the certified key pair (for personal or client certificates)—to form this type of DN, use the CN component to specify the user's full name:

```
CN=<user's_full_name>, OU=<user's_division_name>,
O=<company_name>, C=<country_name>
```

For example:

```
CN=Jane Doe, OU=Human Resources, O=Ace Industries, C=US
```

- The server that owns the certified key pair (for SSL server certificates)—to form this type of DN, use the CN component to specify the server's fully qualified host name in the form

```
<machine_name>.<your_domain>.<domain>:
```

```
CN=<host_name>, OU=<division_name>, O=<company_name>,
C=<country_name>
```

For example:

```
CN=cert.netscape.com, OU=Human Resources, O=Ace Industries,
C=US
```

When clients such as Netscape Navigator receive a server certificate, they expect the CN component of the certificate's subject to match the host name in the URL. If the name in the certificate and the host name of the server do not match, Navigator notifies the user and gives the user the choice of not connecting to the server.

For example, if Navigator goes to the URL `https://cert.netscape.com` and receives a certificate from the server, it expects the CN component of the certificate's subject to be `cert.netscape.com`. If the CN component has a different value (for example, `certificate.netscape.com`), Navigator notifies the user that the certificate's subject name does not match the host name in the URL.

## DNs in CA Certificates

In CA certificates issued by Certificate Management System (for both root and subordinate CAs), DNs are used to identify the authority who owns the certified key pair.

To form this type of distinguished name, use the CN component to specify the name of your CA:

CN=<CA\_name>, O=<company\_name>, C=<country\_name>

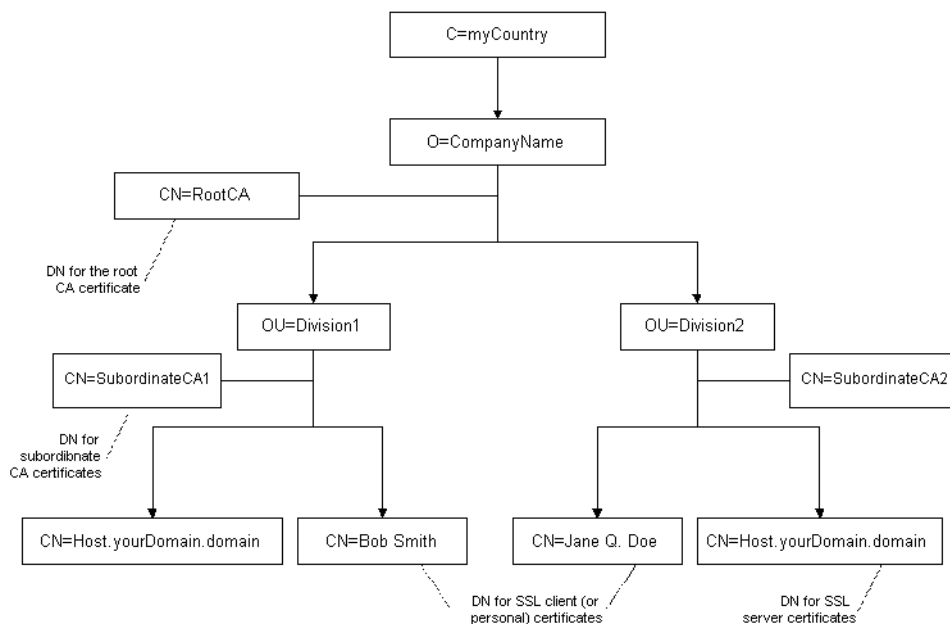
For example:

CN=Ace Industries Certificate Authority, O=Ace Industries,  
C=US

## Selecting DNs for Certificates

Figure 25.4 illustrates the structure of distinguished names you might select for CA certificates, server certificates, and personal certificates.

Figure 25.4 Sample directory hierarchy





## B

# Backing Up and Restoring Data

Each instance of Netscape Certificate Management System (CMS) uses an instance of Netscape Directory Server version 4.x for storing its persistent objects and an ASCII file named `CMS.cfg` for storing its configuration information. This appendix explains how to back up the CMS data and configuration information and how to use the backups to restore data if there is a need.

The appendix has the following sections:

- Before Backing Up and Restoring Data (page 665)
- Backing Up the CMS Configuration and Data (page 668)
- Restoring the CMS Configuration and Data (page 672)

## Before Backing Up and Restoring Data

This section explains the importance of backing up and restoring data and provides guidelines for preparing to use these processes. Read this section only if you are unfamiliar with this means of protecting data.

## What Is a Backup?

An *archive* or *backup* is a copy of all or some portion of the data that Certificate Management System manages; this data is vital to the functioning of the server. More specifically, a backup is a copy of one or more files used by Certificate Management System and any supporting data that you might need in order to restore those files.

**Important** To minimize the chances of a backup being destroyed or getting into the hands of an unauthorized person, be sure to store CMS backups and other backup data in a safe and locked facility. If you need guidelines about securing your backups, see the security measures outlined in the *Netscape Certificate Management System Installation and Deployment Guide*.

## Why You Should Back Up Data

All the information or data needed by Certificate Management System is stored in its internal database (explained in “Configuring the Internal Database” on page 129). If this data becomes corrupt or inaccessible—for example, as a result of program errors, a disk crash, a power outage, or a disaster that damages your entire facility—the server will not function properly. It is therefore strongly recommended that you back up the internal database periodically.

Periodically backing up data will enable you to use the backup for restoring data in the event of data loss.

## Guidelines for Creating a Backup

Here are guidelines to follow in preparation for backing up CMS data:

- Plan your backup schedule.
- Choose the appropriate backup media.
- Check the size of the internal database and configuration files, and make sure that the media has sufficient storage space for creating your backup.
- Decide whether you want to back up the internal database and configuration files to separate media. For example, if you plan to back up the internal database to another directory, you will have to copy the configuration files to another media.

- Verify data consistency.
- Use the appropriate server modes for both Certificate Management System and Netscape Directory Server.
- Estimate the time required for a backup, and ensure that an operator is available.
- If the backup media is tape, label the tapes appropriately.

## What Is a Restore?

The process of restoring data from backups in the event of a data loss is called a *restore*. During a restore, you copy the data from the backup medium to the internal database of Certificate Management System.

## When to Restore Data

If the data managed by Certificate Management System becomes corrupt or inaccessible (for example, as a result of program errors, a disk crash, or a disaster that damages your entire facility), the server will not function properly. To restore Certificate Management System to its original state, you need to use the data backups that you created.

## Guidelines for Restoring Data

Here are some guidelines that will help you restore your Certificate Management System internal database:

- Choose the appropriate backup media.
- Decide on the type of restore.
- Verify your internal database configuration and schema.
- Perform the restore.

# Backing Up the CMS Configuration and Data

Backing up Certificate Management System involves the following steps:

- Step 1. Back Up the Configuration Files
- Step 2: Back Up the Key Pairs
- Step 3. Back Up the Internal Database

**Important** The procedure below explains in general how to back up the server. Be sure to check the following site for technical notes on backing up and restoring Certificate Management System:

<http://home.netscape.com/eng/server/cms>

## Step 1. Back Up the Configuration Files

You can back up the configuration information pertaining to a CMS instance to any backup media, such as a tape or the file system of a computer designated for backing up important files. The following procedure explains how to back up the configuration information of a CMS instance to the file system of a machine used for backups. If your backup media is a tape or any other backup unit, follow the instructions that came with it.

To back up the configuration information of a CMS instance:

1. Shut down the CMS instance whose configuration information you want to back up; see “Stopping Certificate Management System” on page 110.
2. In the file system of the machine used for backups, create a back up directory for storing the configuration information.
3. Copy the complete `config` directory contents.

You can find this directory at this location:

```
<server_root>/cert-<instance_id>/...
```

<server\_root> is the directory where the CMS binaries are kept. You first specified this directory during installation.

<instance\_id> is the ID for this instance of Certificate Management System. You first specified this when you installed this server.

## Step 2: Back Up the Key Pairs

Because the destruction of a private key in a disk crash or similar event can be disastrous if you are depending upon that key for a hierarchy of certificate authorities, backing up your key data is commensurately important. If you do make copies of your keys, however, you must protect your backups with the same level of security that you use for protecting your original keys.

If you generated your server's key pairs using the internal token, the key pairs are stored in a file named `key3.db` file; this is one of the files in the configuration directory. When you backed up the configuration directory in Step 1, this file was copied to the backup media. Make sure that only you or authorized administrators have access to the backup media—for example, if you copied the configuration directory to a backup machine, make sure that the machine is in a locked facility and that it has restricted access.

If the keys are in an external token, such as a smart card, keep it in a locked facility.

## Step 3. Back Up the Internal Database

You can back up the internal database in the following ways:

- By using the LDIF conversion built into the Directory Server window
- By using the `db2bak` command-line script
- By manually copying the internal database directly to a backup directory

The following procedure explains how to back up the internal database using its administration interface, called the *Directory Server window*, in Netscape Console. When you back up your database from the Directory Server window,

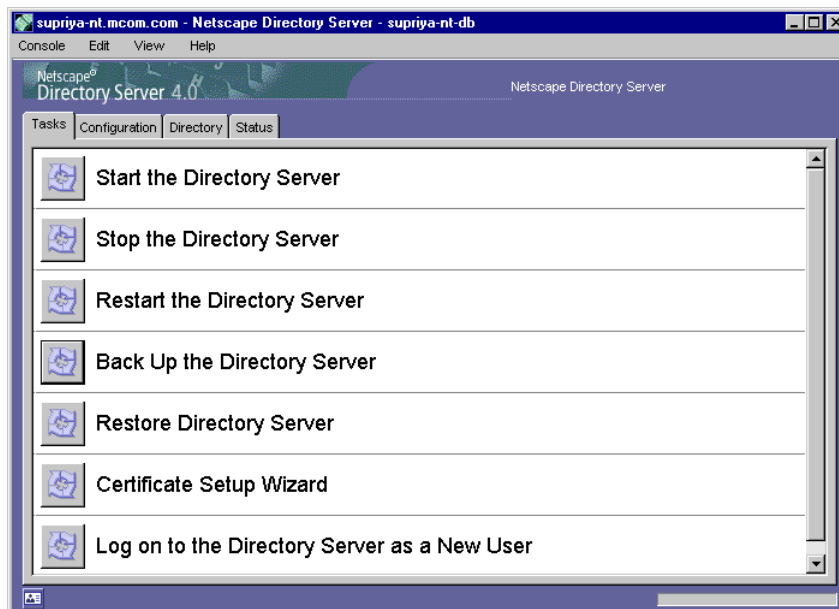
the server copies the entire database and associated index files to a backup location. For backing up the internal database by using other methods, see the Netscape Directory Server 4.x documentation.

You can back up the internal database online or offline—that is, while the server is running or shut down.

To back up the internal database of a CMS instance:

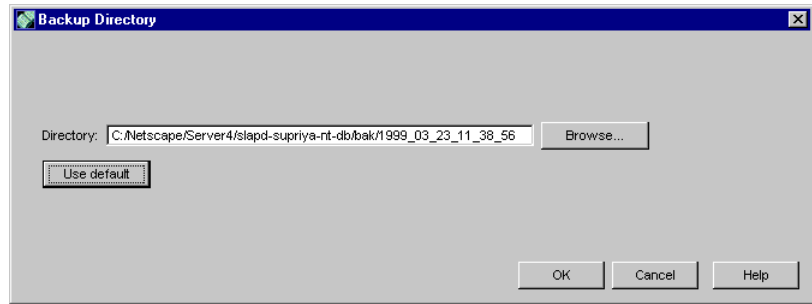
1. In Netscape Console, locate the internal database instance you want to back up.
2. Select the instance and click Open.

The Directory Server window appears with the Tasks tab open.



3. Click “Back Up the Directory Server.”

The Backup Directory dialog box appears.



4. Choose a directory name where you want the backup stored.

You can either choose a new directory or use the default directory that the server provides.

**Directory.** Type the full path to the directory in which you want the server to store the backup file, or click the Browse button to select an existing directory.

**Use default.** Click this button if you want the server to suggest a path for you. If you choose this option, the server stores the backup file in this location:

```
<server_root>/slapd-<cms_instance_id>/bak/<backup_name>
```

<server\_root> is the directory where the binaries for this instance of Certificate Management System are kept. You first specified this directory during installation.

<cms\_instance\_id> is the ID for this instance of Certificate Management System. You first specified this when you installed this server.

bak is the directory to which files are backed up.

<backup\_name> is a directory-specified name for the backup. By default, the backup name identifies the date and time when the backup was created in the format YYYY\_MM\_DD\_HH\_mm\_SS; the date and time has YYYYMMDD (year, month, day) and HHmmSS (hour, minute, second) forms, in that order. For example, a backup created on March 23, 1999 at 11:38:56 a.m. is named 1999\_03\_23\_11\_38\_56.

5. Click OK.

The backup process begins.

## Restoring the CMS Configuration and Data

If the CMS internal database is corrupted, you can restore it from a previously generated backup. The restore process consists of copying the configuration files and the Directory Server and its associated index files from the backup location to the internal database of Certificate Management System.

Note that restoring your CMS internal database overwrites any existing files.

**Note** Be sure to check the following site for technical notes on backing up and restoring Certificate Management System:

`http://home.netscape.com/eng/server/cms`





# Command-Line Utilities

Netscape Certificate Management System (CMS) is bundled with various command-line utilities. This appendix summarizes these utilities, explains a few of them, and provides pointers for the rest.

The appendix has the following sections:

- Summary of Command-Line Utilities (page 673)
- ASCII to Binary Tool (page 676)
- Binary to ASCII Tool (page 677)
- Pretty Print Certificate Tool (page 678)
- Pretty Print CRL Tool (page 681)
- dumpasn1 Tool (page 683)

## Summary of Command-Line Utilities

Table 25.2 summarizes the various command-line utilities that are bundled with Certificate Management System.

Table 25.2 Summary of command-line utilities

Utility	Description
<code>AtOB</code> (ASCII to Binary Tool)	Converts ASCII base-64 encoded data to binary base-64 encoded data. For details, see “ASCII to Binary Tool” on page 676.
<code>BtoA</code> (Binary to ASCII Tool)	Converts binary base-64 encoded data to ASCII base-64 encoded data. For details, see “Binary to ASCII Tool” on page 677.
<code>PrettyPrintCert</code> (Pretty Print Certificate Tool)	Prints the contents of a certificate stored as ASCII base-64 encoded data in a human-readable form. For details, see “Pretty Print Certificate Tool” on page 678.
<code>PrettyPrintCrl</code> (Pretty Print CRL Tool)	Prints the contents of a CRL stored as ASCII base-64 encoded data in a human-readable form. For details, see “Pretty Print CRL Tool” on page 681.
<code>dumpasn1</code>	Dumps the contents of binary base-64-encoded data. For details, see “dumpasn1 Tool” on page 683.
<code>certutil</code> (Certificate Database Tool)	View and manipulate the certificate database ( <code>cert7.db</code> ) contents. For details, see “Certificate Database Tool” on page 685.
<code>keyutil</code> (Key Database Tool)	View and manipulate the key database ( <code>key3.db</code> ) contents. For details, see “Key Database Tool” on page 701.
<code>signtool</code> (Netscape Signing Tool)	Used to digitally sign any file, including log files. For details, see “Netscape Signing Tool” on page 711.

Table 25.2 Summary of command-line utilities (Continued)

Utility	Description
<code>sslstrength</code> (SSL Strength Tool)	Used to connect to an SSL server and report back the type and strength of the encryption cipher that it's using. For details, see "SSL Strength Tool" on page 745.
<code>ssltap</code> (SSL Debugging Tool)	Used to debug SSL applications. For details, see "SSL Debugging Tool" on page 751.
<code>migrate</code> (Migration Tool)	Migrate data from a Certificate Server 1.x installation into a Certificate Management System installation. For details, see "Appendix A, Migrating from Certificate Server" in the <i>Netscape Certificate Management System Installation and Deployment Guide</i> .
<code>setpin</code> (PIN Generator tool)	Used to generate PINs for end entities for directory plus PIN-based authenticating. For details, see "Using the PIN Generator Tool" on page 285.
<code>killproc</code>	Used to kill system processes in Windows NT. For details, see "Attending to an Unresponsive Server" on page 116.

The first five tools listed in Table 25.2 (`AtoB`, `BtoA`, `PrettyPrintCert`, `PrettyPrintCrl`, and `dumpasn1`) are useful for converting back and forth between various encodings and formats you may encounter when dealing with keys and certificates. (These tools are explained in this appendix.)

The Certificate Database Tool, Key Database Tool, and Security Module Database Tool ("`modutil`" in Appendix B of *Managing Servers with Netscape Console*) are useful for a variety of administrative tasks that involve manipulating certificate and key databases.

The Migration tool is used to convert Certificate Server 1.x data for use with Certificate Management System, and the PIN Generator tool is used to create PINs for directory authentication. The `killproc` tool is used to terminate the Java virtual machines, called `jssjava` processes, when Certificate Management System becomes unresponsive.

The Netscape Signing Tool can be used to associate a digital signature with any file, including CMS log files.

The SSL Strength Tool and SSL Debugging Tool are useful for testing and debugging purposes.

## Location of Command-Line Utilities

Except for the Security Module Database Tool, you can find all the other command-line utilities at this location:

```
<server_root>/bin/cert/tools/...
```

`<server_root>` is the directory where the CMS binaries are kept. You first specified this directory during installation.

The Security Module Database Tool (which is explained in "modutil" in Appendix B of *Managing Servers with Netscape Console*), is located here:

```
<server_root>/shared/bin/modutil
```

## ASCII to Binary Tool

You can use the ASCII to Binary tool to convert ASCII base-64 encoded data to binary base-64 encoded data.

## Availability

This tool is available for Solaris 2.5.1 (SunOS 5.5.1), Solaris 2.6 (SunOS 5.6), HP-UX B.11.00, AIX 4.2, and Windows NT 4.0.

## Syntax

To run the ASCII to Binary tool, type the following command:

```
AtoB[.bat] <input_file> <output_file>
```

`.bat` specifies the file extension; this is required only when running the utility on a Windows NT system.

`<input_file>` specifies the path to the file that contains the base-64 encoded data in ASCII format.

`<output_file>` specifies the path to the file to write the base-64 encoded data in binary format.

## Example

```
AtoB.bat C:\test\data.in C:\test\data.out
```

The above command takes the base-64 encoded data (in ASCII format) in the file named `data.in` and writes the binary equivalent of the data to the file named `data.out`.

# Binary to ASCII Tool

You can use the Binary to ASCII tool to convert binary base-64 encoded data to ASCII base-64 encoded data.

## Availability

This tool is available for Solaris 2.5.1 (SunOS 5.5.1), Solaris 2.6 (SunOS 5.6), HP-UX B.11.00, AIX 4.2, and Windows NT 4.0.

## Syntax

To run the Binary to ASCII tool, type the following command:

```
BtoA[.bat] <input-file> <output_file>
```

.bat specifies the file extension; this is required only when running the utility on a Windows NT system.

<input\_file> specifies the path to the file that contains the base-64 encoded data in binary format.

<output\_file> specifies the path to the file to write the base-64 encoded data in ASCII format.

## Example

```
BtoA.bat C:\test\data.in C:\test\data.out
```

The above command takes the base-64 encoded data (in binary format) in the file named data.in and writes the ASCII equivalent of the data to the file named data.out.

# Pretty Print Certificate Tool

You can use the Pretty Print Certificate tool to print the contents of a certificate stored as ASCII base-64 encoded data in a human-readable form.

## Availability

This tool is available for Solaris 2.5.1 (SunOS 5.5.1), Solaris 2.6 (SunOS 5.6), HP-UX B.11.00, AIX 4.2, and Windows NT 4.0.

## Syntax

To run the Pretty Print Certificate tool, type the following command:

```
PrettyPrintCert[.bat] <input_file> [<output_file>]
```

.bat specifies the file extension; this is required only when running the utility on a Windows NT system.

`<input_file>` specifies the path to the file that contains the base-64 encoded certificate.

`<output_file>` specifies the path to the file to write the certificate. This argument is optional; if you don't specify an output file, the certificate information is written to the standard output.

## Example

```
PrettyPrintCert.bat C:\test\cert.in C:\test\cert.out
```

The above command takes the base-64 encoded certificate in the `cert.in` file and writes the certificate in the pretty-print form to the output file named `cert.out`.

The base-64 encoded certificate (content of the `cert.in` file) would look similar to this:

[illegible]

The certificate in pretty-print form (content of the `cert.out` file) would look similar to this:

```
Certificate:

    Data:

        Version:  v3

        Serial Number:  0x100C

        Signature Algorithm:  OID.1.2.840.113549.1.1.5 -
1.2.840.113549.1.1.5

        Issuer:  CN=Test Test CA,OU=Widget Makers 'R'Us,O=PalookaVille
Widgets\, Inc.,C=US

        Validity:

            Not Before:  Wednesday, February 17, 1999 7:43:39 PM

            Not After:   Thursday, February 17, 2000 7:43:39 PM
```

Subject: MAIL=admin@netscape.com,CN=testCA,Administrator  
UID=admin,OU=Netscape CMS,O=Netscape Comm Corp.,C=US

Subject Public Key Info:

Algorithm: RSA - 1.2.840.113549.1.1.1

Public Key:

30:81:89:02:81:81:00:DE:26:B3:C2:9D:3F:7F:FA:DF:  
24:E3:9B:7A:24:AC:89:AD:C1:BA:27:D1:1C:13:70:F7:  
96:59:41:1F:4D:21:7A:F5:C7:96:C4:75:83:35:9F:49:  
E4:B0:A7:5F:95:C4:09:EA:67:00:EF:BD:7C:39:92:11:  
31:F2:CA:C9:16:87:B9:AD:B8:39:69:18:CE:29:81:5F:  
F3:4D:97:B9:DF:B7:60:B3:00:03:16:8E:C1:F8:17:6E:  
7A:D2:00:0F:7D:9B:A2:69:35:18:70:1C:7C:AE:12:2F:  
0B:0F:EC:69:CD:57:6F:85:F3:3E:9D:43:64:EF:0D:5F:  
EF:40:FF:A6:68:FD:DD:02:03:01:00:01:

Extensions:

Identifier: 2.16.840.1.113730.1.1

Critical: no

Value:

03:02:00:A0:

Identifier: Authority Key Identifier - 2.5.29.35

Critical: no

Key Identifier:

EB:B5:11:8F:00:9A:1A:A6:6E:52:94:A9:74:BC:65:CF:  
07:89:2A:23:

Signature:

Algorithm: OID.1.2.840.113549.1.1.5 - 1.2.840.113549.1.1.5

Signature:

3E:8A:A9:9B:D1:71:EE:37:0D:1F:A0:C1:00:17:53:26:  
6F:EE:28:15:20:74:F6:C5:4F:B4:E7:95:3C:A2:6A:74:  
92:3C:07:A8:39:12:1B:7E:C4:C7:AE:79:C8:D8:FF:1F:  
D5:48:D8:2E:DD:87:88:69:D5:3A:06:CA:CA:9C:9A:55:  
DA:A9:E8:BF:36:BC:68:6D:1F:2B:1C:26:62:7C:75:27:  
E2:8D:24:4A:14:9C:92:C6:F0:7A:05:A1:52:D7:CC:7D:  
E0:9D:6C:D8:97:3A:9C:12:8C:25:48:7F:51:59:BE:3C:  
2B:30:BF:EB:0A:45:7D:A6:49:FB:E7:BE:04:05:D6:8F:



# Pretty Print CRL Tool

You can use the Pretty Print CRL tool to print the contents of a CRL stored as ASCII base-64-encoded data in a human-readable form.

## Availability

This tool is available for Solaris 2.5.1 (SunOS 5.5.1), Solaris 2.6 (SunOS 5.6), HP-UX B.11.00, AIX 4.2, and Windows NT 4.0.

## Syntax

To run the Pretty Print CRL tool, type the following command:

```
PrettyPrintCrl[.bat] <input_file> [<output-file>]
```

`.bat` specifies the file extension; this is required only when running the utility on a Windows NT system.

`<input_file>` specifies the path to the file that contains the base-64 encoded CRL.

`<output_file>` specifies the path to the file to write the CRL. This argument is optional; if you don't specify an output file, the CRL information is written to the standard output.

## Example

```
PrettyPrintCrl.bat C:\test\crl.in C:\test\crl.out
```

The above command takes the base-64 encoded CRL in the `crl.in` file and writes the CRL in the pretty-print form to the output file named `crl.out`.

The base-64 encoded CRL (content of the `crl.in` file) would look similar to this:

```
-----BEGIN CRL-----
```

```

MIIBKjCBAIBATANBgkqhkiG9w0BAQQFADAsMREwDwYDVQQKEwhOZXRzY2FwZTEuMBUGA1UEAxMOQ2
VydDQwIFRlc3QgQ0EXDTk4MTIxNzIyMzcyNFowgaowIAIBExcNOTgxMjE1MTMxODMyWjAMMAoGA1U
dFQQCgEBMCACARIXDTk4MTIxNTEzMjA0MlowDDAKBgNVHRUEAwOBAjAgAgERFw05ODEyMTYxMjUx
NTRaMAAwCgYDVROVBAMKAQEwIAIBEBcNOTgxMjE3MTAzNzI0WjAMMAoGA1UdFQQCgEDMCACAQoXD
Tk4MTEyNTEzMTEzOFowDDAKBgNVHRUEAwOBTANBgkqhkiG9w0BAQQFAAOBgQBCN8500GPTnHfImY
PROvoorx7HyFz2ZsuKsVblTcemsX0NL7DtOa+MyY0pPrkXgm157JrkxEJ7GB0eogbAS6iFbmeSqPH
j8+JBH5stJNnfTCuhaM6Wx63Wc9LwZXOXTpsvpGxq0YYI0+DPfBZlI3z4lCsNczxJV+9NkeMrheEg
==
-----END CRL-----

```

The CRL in pretty-print form (content of the `cr1.out` file) would look similar to this:

#### Certificate Revocation List:

##### Data:

Version: v2

Signature Algorithm: MD5withRSA - 1.2.840.113549.1.1.4

Issuer: CN=Cert40 Test CA,O=Netscape

This Update: Thu Dec 17 14:37:24 PST 1998

##### Revoked Certificates:

Serial Number: 0x13

Revocation Date: Tuesday, December 15, 1998 5:18:32 AM

##### Extensions:

Identifier: Revocation Reason - 2.5.29.21

Critical: no

Reason: Key\_Compromise

Serial Number: 0x12

Revocation Date: Tuesday, December 15, 1998 5:20:42 AM

##### Extensions:

Identifier: Revocation Reason - 2.5.29.21

Critical: no

Reason: CA\_Compromise

Serial Number: 0x11

Revocation Date: Wednesday, December 16, 1998 4:51:54 AM

##### Extensions:

Identifier: Revocation Reason - 2.5.29.21

Critical: no

Reason: Key\_Compromise

Serial Number: 0x10

Revocation Date: Thursday, December 17, 1998 2:37:24 AM

##### Extensions:

```

Identifier: Revocation Reason - 2.5.29.21
Critical: no
Reason: Affiliation_Changed
Serial Number: 0xA
Revocation Date: Wednesday, November 25, 1998 5:11:18 AM
Extensions:
Identifier: Revocation Reason - 2.5.29.21
Critical: no
Reason: Key_Compromise
Signature:
Algorithm: MD5withRSA - 1.2.840.113549.1.1.4
Signature:
42:37:CE:4E:D0:63:D3:9C:77:C8:99:83:D1:3A:FA:28:
AF:1E:C7:C8:5C:F6:66:CB:8A:B1:56:E5:4D:C7:A6:B1:
7D:0D:2F:B0:ED:39:AF:8C:C9:8D:29:3E:B9:17:82:6D:
79:EC:9A:E4:C4:42:7B:18:13:9E:A2:06:C0:4B:A8:85:
6E:67:92:A8:F1:E3:F3:E2:41:1F:9B:2D:24:D9:DF:4C:
2B:A1:68:CE:96:C7:AF:F7:5B:F7:3D:2F:06:57:39:74:
CF:B2:FA:46:C6:AD:18:60:8D:3E:0C:F7:C1:66:52:37:
CF:89:42:B0:D7:33:C4:95:7E:F4:D9:1E:32:B8:5E:12:

```

## dumpasn1 Tool

The dumpasn1 tool is freeware that is packaged with Certificate Management System for your convenience. You can use this tool to dump the contents of binary base-64 encoded data. For more information about this tool, see this URL:

<http://www.cs.auckland.ac.nz/~pgut001/>



## D

# Certificate Database Tool

Certificate Database Tool is a command-line utility that can create the certificate database file (`cert7.db`) for Certificate Management System. The utility can also list, generate, modify, or delete certificates within the file.

Certificate database management tasks are part of a process that typically also involves managing key databases (`key3.db` files). The key and certificate management process generally begins with creating keys in the key database, then generating and managing certificates in the certificate database.

This appendix discusses certificate database management:

- Availability (page 686)
- Syntax (page 686)
- Usage (page 694)
- Examples (page 695)

**Note** For information on key database and security module database management, see “Key Database Tool” on page 701 and “modutil” in Appendix B of *Managing Servers with Netscape Console*.

# Availability

This tool is available for Solaris 2.5.1 (SunOS 5.5.1) and Windows NT 4.0.

## Syntax

To run Certificate Database Tool, type the following command:

```
certutil option [arguments]
```

where *options* and *arguments* are combinations of the options and arguments listed in the following section. Each command takes one option. Each option may take zero or more arguments. To see a usage string, issue the command without options, or with the `-H` option.

## Options and Arguments

Options specify an action and are uppercase. Option arguments modify an action and are lowercase. Certificate Database Tool command options and their arguments are defined as follows:

---

### Options

<code>-N</code>	Create a new certificate database.
<code>-S</code>	Create an individual certificate and add it to a certificate database.
<code>-R</code>	Create a certificate-request file that can be submitted to a certificate authority (CA) for processing into a finished certificate. Output defaults to standard out unless you use <code>-o output-file</code> argument.
	Use the <code>-a</code> argument to specify ASCII output.

---

---

-C	Create a new binary certificate file from a binary certificate-request file. Use the <code>-i</code> argument to specify the certificate-request file. If this argument is not used Certificate Database Tool prompts for a filename.
-A	Add an existing certificate to a certificate database. The certificate database should already exist; if one is not present, this option will initialize one by default.
-L	List all the certificates, or display information about a named certificate, in a certificate database.  Use the <code>-h <i>tokenname</i></code> argument to specify the certificate database on a particular hardware or software token.
-V	Check the validity of a certificate and its attributes.
-M	Modify a certificate's trust attributes using the values of the <code>-t</code> argument.
-H	Display a list of the options and arguments used by Certificate Database Tool.

### Arguments

-a	Use ASCII format or allow the use of ASCII format for input or output. This formatting follows RFC #1113. For certificate requests, ASCII output defaults to standard output unless redirected.
----	---

---

---

<code>-b <i>validity-time</i></code>	<p>Specify a time at which a certificate is required to be valid. Use when checking certificate validity with the <code>-V</code> option. The format of the <i>validity-time</i> argument is "YYMMDDHHMMSS[+HHMM -HHMM Z]". Specifying seconds (SS) is optional. When specifying an explicit time, use "YYMMDDHHMMSSZ". When specifying an offset time, use "YYMMDDHHMMSS+HHMM" or "YYMMDDHHMMSS-HHMM". If this option is not used, the validity check defaults to the current system time.</p>
<code>-c <i>issuer</i></code>	<p>Identify the certificate of the CA from which a new certificate will derive its authenticity. Use the exact nickname or alias of the CA certificate, or use the CA's email address. Bracket the <i>issuer</i> string with quotation marks if it contains spaces.</p>
<code>-d <i>certdir</i></code>	<p>Specify a directory containing a certificate database file. On Unix Certificate Database Tool defaults to <code>\$HOME/.netscape</code> (that is, <code>~/netscape</code>). On Windows NT the default is the current directory.</p> <p>The <code>cert7.db</code> and <code>key3.db</code> database files must reside in the same directory.</p>
<code>-e</code>	<p>Check a certificate's signature during the process of validating a certificate.</p>
<code>-f <i>password-file</i></code>	<p>Specify a file that will automatically supply the password to include in a certificate or to access a certificate database. This is a plain-text file containing one password. Be sure to prevent unauthorized access to this file.</p>

---



---

<code>-h <i>tokenname</i></code>	Specify the name of a token to use or act on. Unless specified otherwise the default token is an internal slot (specifically, internal slot 2). This slot can also be explicitly named with the string "internal". An internal slots is a virtual slot maintained in software, rather than a hardware device. Internal slot 2 is used by key and certificate services. Internal slot 1 is used by cryptographic services.
<code>-i <i>cert</i>   <i>cert-request-file</i></code>	Specify a specific certificate, or a certificate-request file.
<code>-k <i>shortkeyID</i></code>	Specify the public key to use when creating a certificate or certificate request. The <i>shortkeyID</i> is the first few bytes of the keyID (as shown by the <code>keyutil -L</code> command), starting from the second byte, with a length sufficient to identify it uniquely.
<code>-l</code>	Display detailed information when validating a certificate with the <code>-V</code> option.
<code>-m <i>serial-number</i></code>	Assign a unique serial number to a certificate being created. This operation should be performed by a CA. The default serial number is 0 (zero). Serial numbers are limited to integers.
<code>-n <i>certname</i></code>	Specify the nickname of a certificate to list, create, add to a database, modify, or validate. Bracket the <i>certname</i> string with quotation marks if it contains spaces.
<code>-o <i>output-file</i></code>	Specify the output file name for new certificates or binary certificate requests. Bracket the <i>output-file</i> string with quotation marks if it contains spaces. If this argument is not used the output destination defaults to standard output.

---

---

<code>-p <i>phone</i></code>	Specify a contact telephone number to include in new certificates or certificate requests. Bracket this string with quotation marks if it contains spaces.
<code>-r</code>	Display a certificate's binary DER encoding when listing information about that certificate with the <code>-L</code> option.
<code>-s <i>subject</i></code>	Identify a particular certificate owner for new certificates or certificate requests. Bracket this string with quotation marks if it contains spaces. The subject identification format follows RFC #1485.

---

---

**-t** *trustargs*

Specify the trust attributes to modify in an existing certificate or to apply to a certificate when creating it or adding it to a database.

There are three available trust categories for each certificate, expressed in this order: "*SSL, email, object signing*". In each category position use zero or more of the following attribute codes:

- p** Valid peer
- P** Trusted peer (implies **p**)
- c** Valid CA
- T** Trusted CA to issue client certificates (implies **c**)
- C** Trusted CA to issue server certificates (SSL only) (implies **c**)
- u** Certificate can be used for authentication or signing
- w** Send warning (use with other attributes to include a warning when the certificate is used in that context)

The attribute codes for the categories are separated by commas, and the entire set of attributes enclosed by quotation marks. For example:

**-t** "TCu,Cu,Tuw"

Use the **-L** option to see a list of the current certificates and trust attributes in a certificate database.

**-u** *certusage*

Specify a usage context to apply when validating a certificate with the **-V** option. The contexts are the following:

- C** (as an SSL client)
  - V** (as an SSL server)
  - S** (as an email signer)
  - R** (as an email recipient)
-

---

<code>-v <i>valid-months</i></code>	Set the number of months a new certificate will be valid. The validity period begins at the current system time unless an offset is added or subtracted with the <code>-w</code> option. If this argument is not used, the default validity period is three months. When this argument is used, the default three-month period is automatically added to any value given in the <i>valid-month</i> argument. For example, using this option to set a value of 3 would cause 3 to be added to the three-month default, creating a validity period of six months. You can use negative values to reduce the default period. For example, setting a value of -2 would subtract 2 from the default and create a validity period of one month.
<code>-w <i>offset-months</i></code>	Set an offset from the current system time, in months, for the beginning of a certificate's validity period. Use when creating the certificate or adding it to a database. Express the offset in integers, using a minus sign (-) to indicate a negative offset. If this argument is not used, the validity period begins at the current system time. The length of the validity period is set with the <code>-v</code> argument.
<code>-x</code>	Use Certificate Database Tool to generate the signature for a certificate being created or added to a database, rather than obtaining a signature from a separate CA.
<code>-y <i>rsa dsa</i></code>	Specify the type of key used to generate a new certificate, either RSA or DSA. The default is <i>rsa</i> .

---

- 
- |    |   |
|----|---|
| -1 | Add a key usage extension to a certificate that is being created or added to a database. This extension allows a certificate's key to be dedicated to supporting specific operations such as SSL server or object signing. Certificate Database Tool will prompt you to select a particular usage for the certificate's key. These usages are described under "Standard X.509 v3 Certificate Extensions" in Appendix C of <i>Netscape Certificate Management System Installation and Deployment Guide</i> .   |
| -2 | Add a basic constraint extension to a certificate that is being created or added to a database. This extension supports the certificate chain verification process. Certificate Database Tool will prompt you to select the certificate constraint extension. Constraint extensions are described in "Standard X.509 v3 Certificate Extensions" in Appendix C of <i>Netscape Certificate Management System Installation and Deployment Guide</i> .  |
| -3 | Add an authority key ID extension to a certificate that is being created or added to a database. This extension supports the identification of a particular certificate, from among multiple certificates associated with one subject name, as the correct issuer of a certificate. Certificate Database Tool will prompt you to select the authority key ID extension. Authority key ID extensions are described under "Standard X.509 v3 Certificate Extensions" in Appendix C of <i>Netscape Certificate Management System Installation and Deployment Guide</i> . |
| -4 | Add a CRL distribution point extension to a certificate that is being created or added to a database. This extension identifies the URL of a certificate's associated certificate revocation list (CRL). Certificate Database Tool prompts you to enter the URL.  |
-

# Usage

Certificate Database Tool's capabilities are grouped as follows, using these combinations of options and arguments. Options and arguments in square brackets are optional, those without square brackets are required.

- Creating a new `cert7.db` file:

```
-N [-d certdir]
```

- Creating a new certificate and adding it to the database with one command:

```
-S -k shortkeyID -y rsa|dsa -n certname -s subject  
[-c issuer [-x] -t trustargs [-h tokenname]  
[-m serial-number] [-v valid-months] [-w offset-months]  
[-d certdir] [-p phone] [-f password-file] [-1] [-2] [-3] [-4]
```

- Making a separate certificate request:

```
-R -k shortkeyID -y rsa|dsa -s subject [-h tokenname]  
[-d certdir] [-p phone] [-o output-file] [-f password-file]
```

- Creating a new binary certificate from a binary certificate request:

```
-C [-c issuer [-k shortkeyID -y rsa|dsa -x] [-f password-file]  
[-h tokenname] -i cert-request-file -o output-file [-m serial-number]  
[-v valid-months] [-w offset-months] [-d certdir] [-1] [-2] [-3]  
[-4]
```

- Adding a certificate to an existing database:

```
-A -n certname -t trustargs [-h tokenname] [-d certdir] [-a]  
[-i cert-request-file]
```

- Listing all certificates or a named certificate:

```
-L [-n certname] [-d certdir] [-r] [-a]
```

- Validating a certificate:

```
-V -n certname -b validity-time -u certusage [-e] [-1] [-d certdir]
```

- Modifying a certificate's trust attribute:

```
-M -n certname -t trustargs [-d certdir]
```

- Displaying a list of the options and arguments used by Certificate Database Tool:  
-H

## Examples

This section contains examples for the following tasks:

- Creating a New Certificate Database
- Listing Certificates in a Database
- Creating a Certificate Request
- Creating a Certificate
- Adding a Certificate to the Database
- Validating a Certificate

### Creating a New Certificate Database

This example creates a new certificate database (`cert7.db` file) in the specified directory:

```
certutil -N -d certdir
```

You must generate the associated `key3.db` and `secmod.db` files by using the Key Database Tool or other tools.

### Listing Certificates in a Database

This example lists all the certificates in the `cert7.db` file in the specified directory:

```
certutil -L -d certdir
```

Certificate Database Tool displays output similar to the following:

Certificate Name	Trust Attributes
Uptime Group Plc. Class 1 CA	C,C,
VeriSign Class 1 Primary CA	,C,
VeriSign Class 2 Primary CA	C,C,C
AT&T Certificate Services	C,C,
GTE CyberTrust Secure Server CA	C,,
Verisign/RSA Commercial CA	C,C,
AT&T Directory Services	C,C,
BelSign Secure Server CA	C,,
Verisign/RSA Secure Server CA	C,C,
GTE CyberTrust Root CA	C,C,
Uptime Group Plc. Class 4 CA	,C,
VeriSign Class 3 Primary CA	C,C,C
Canada Post Corporation CA	C,C,
Integrion CA	C,C,C
IBM World Registry CA	C,C,C
GTIS/PWGSC, Canada Gov. Web CA	C,C,
GTIS/PWGSC, Canada Gov. Secure CA	C,C,C
MCI Mall CA	C,C,
VeriSign Class 4 Primary CA	C,C,C
KEYWITNESS, Canada CA	C,C,
BelSign Object Publishing CA	,,C
BBN Certificate Services CA Root 1	C,C,
p Valid peer	
P Trusted peer (implies p)	
c Valid CA	
T Trusted CA to issue client certs (implies c)	
C Trusted CA to issue server certs(for ssl only) (implies c)	
u User cert	
w Send warning	

## Creating a Certificate Request

This example generates a binary certificate request file named `e95c.req` in the specified directory:

```
certutil -R -s "CN=John Smith, O=Netscape, L=Mountain View,
ST=California, C=US" -p "650-555-8888" -k e95c -o e95c.req -d
certdir
```



Before it creates the request file, Certificate Database Tool prompts you for a password:

```
Enter Password or Pin for "Communicator Certificate DB":
```

## Creating a Certificate

A valid certificate must be issued by a trusted CA. If a CA key pair is not available, you can create a self-signed certificate (for purposes of illustration) with the `-x` argument. This example creates a new, self-signed binary certificate named `e95c.crt`, from a binary certificate request named `e95c.req`, in the specified directory.

```
certutil -C -i e95c.req -o e95c.crt -k e95c -m 1234
-f password-file -x -d certdir
```

The following example creates a new binary certificate named `one.crt`, from a binary certificate request named `one.req`, in the specified directory. It is issued by the self-signed certificate created above, `e95c.crt`.

```
certutil -C -m 2345 -i one.req -o one.crt -c e95c.crt -d certdir
```

## Adding a Certificate to the Database

This example adds a certificate to the certificate database:

```
certutil -A -n jsmith@netscape.com -t "C,C,C" -i e95c.crt
-d certdir
```

You can see this certificate in the database with this command:

```
certutil -L -n jsmith@netscape.com -d certdir
```

Certificate Database Tool displays output similar to the following:

```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 0 (0x0)
    Signature Algorithm: PKCS #1 MD5 With RSA Encryption
    Issuer: CN=John Smith, O=Netscape, L=Mountain View,
```

```

ST=California, C=US
  Validity:
    Not Before: Thu Mar 12 00:10:40 1998
    Not After: Sat Sep 12 00:10:40 1998
Subject: CN=John Smith, O=Netscape, L=Mountain View,
ST=California, C=US

Subject Public Key Info:
  Public Key Algorithm: PKCS #1 RSA Encryption
  RSA Public Key:
    Modulus:
      00:da:53:23:58:00:91:6a:d1:a2:39:26:2f:06:3a:
      38:eb:d4:c1:54:a3:62:00:b9:f0:7f:d6:00:76:aa:
      18:da:6b:79:71:5b:d9:8a:82:24:07:ed:49:5b:33:
      bf:c5:79:7c:f6:22:a7:18:66:9f:ab:2d:33:03:ec:
      63:eb:9d:0d:02:1b:da:32:ae:6c:d4:40:95:9f:b3:
      44:8b:8e:8e:a3:ae:ad:08:38:4f:2e:53:e9:e1:3f:
      8e:43:7f:51:61:b9:0f:f3:a6:25:1e:0b:93:74:8f:
      c6:13:a3:cd:51:40:84:0e:79:ea:b7:6b:d1:cc:6b:
      78:d0:5d:da:be:2b:57:c2:6f
    Exponent: 65537 (0x10001)
  Signature Algorithm: PKCS #1 MD5 With RSA Encryption
  Signature:
    44:15:e5:ae:c4:30:2c:cd:60:89:f1:1d:22:ed:5e:5b:10:c8:
    7e:5f:56:8c:b4:00:12:ed:5f:a4:6a:12:c3:0d:01:03:09:f2:
    2f:e7:fd:95:25:47:80:ea:c1:25:5a:33:98:16:52:78:24:80:
    c9:53:11:40:99:f5:bd:b8:e9:35:0e:5d:3e:38:6a:5c:10:d1:
    c6:f9:54:af:28:56:62:f4:2f:b3:9b:50:e1:c3:a2:ba:27:ee:
    07:9f:89:2e:78:5c:6d:46:b6:5e:99:de:e6:9d:eb:d9:ff:b2:
    5f:c6:f6:c6:52:4a:d4:67:be:8d:fc:dd:52:51:8e:a2:d7:15:
    71:3e

Certificate Trust Flags:
  SSL Flags:
    Valid CA
    Trusted CA
  Email Flags:
    Valid CA
    Trusted CA
  Object Signing Flags:
    Valid CA
    Trusted CA

```

## Validating a Certificate

This example validates a certificate:

```
certutil -V -n jsmith@netscape.com -b 9803201212Z -u SR -e -l  
-d certdir
```

Certificate Database Tool shows results similar to

```
Certificate:'jsmith@netscape.com' is valid.
```

or

```
UID=jsmith, E=jsmith@netscape.com, CN=John Smith, O=Netscape  
Communications Corp., C=US : Expired certificate
```

or

```
UID=jsmith, E=jsmith@netscape.com, CN=John Smith, O=Netscape  
Communications Corp., C=US : Certificate not approved for this  
operation
```



## E

# Key Database Tool

Key Database Tool is a command-line utility that can modify the key database file (`key3.db`) of Netscape Certificate Management System (CMS). You can use the utility to create or change the database password, generate new public and private key pairs, display the contents of the database, or delete key pairs from the database.

Key database management tasks are part of a process that typically also involves managing client certificate databases (`cert7.db` file). The key and certificate management process generally begins with creating keys in the key database, then generating and managing certificates in the certificate database.

This appendix discusses key database management. For information on certificate database and security module database management, see “Certificate Database Tool” on page 685 and the section “modutil” in Appendix B of *Managing Servers with Netscape Console*.

This appendix has the following sections:

- Availability (page 702)
- Syntax (page 702)
- Usage (page 705)
- Examples (page 706)

# Availability

This tool is available for Solaris 2.5.1 (SunOS 5.5.1) and Windows NT 4.0.

## Syntax

To run Key Database Tool, type the command

```
keyutil option [arguments]
```

where *option* and *arguments* are combinations of the options and arguments listed in the following section. Each command takes one option. Each option may take zero or more arguments. To see a usage string, issue the command without options, or with the `-H` option.

## Options and Arguments

Options specify an action and are uppercase. Option arguments modify an action and are lowercase. Key Database Tool options and their arguments are defined as follows:

---

### Options

-N	Create a new key database and set its password.
	Use the <code>-h <i>tokenname</i></code> argument to specify a specific hardware or software token in which to create the new database.
-C	Change the password to a key database.
-G	Generate a new public and private key pair within a database. The key database should already exist; if one is not present, this option will initialize one by default.
	Some smart cards (for example, the Litronic card) can store only one key pair. If you create a new key pair for such a card, the previous pair is overwritten.

---

---

-L	<p>List the keyID of keys in the key database. A keyID is the modulus of the RSA key or the <code>publicValue</code> of the DSA key. IDs are displayed in hexadecimal (“0x” is not shown).</p> <p>You can identify keys by a <i>shortcutID</i>. The <i>shortcutID</i> is the first few bytes of the keyID, starting from the second byte, with a length sufficient to identify it uniquely.</p> <p>Use the <code>-a</code> argument to list keys of all tokens. Otherwise the list will contain only keys in the default (internal) slot.</p> <p>Use the <code>-l</code> argument to list DSA as well as RSA keys.</p>
-P	Display public key information on the screen.
-D	<p>Delete a private key from a key database. Specify the key to delete with the <code>-k</code> argument. Specify the database from which to delete the key with the <code>-d</code> argument.</p> <p>Use the <code>-t</code> argument to specify explicitly whether to delete a DSA or an RSA key. If you do not use the <code>-t</code> argument, the option looks for an RSA key matching the <i>shortcutID</i>.</p> <p>When you delete keys, be sure to also remove any certificates associated with those keys from the certificate database, by using the Certificate Database Tool.</p> <p>Some smart cards (for example, the Litronic card) do not let you remove a public key you have generated. In such a case, only the private key is deleted from the key pair. You can display the public key with the command <code>keyutil -L -h tokename</code>.</p>
-H	Display a list of the options and arguments used by Key Database Tool.

### Arguments

---

-a	List the RSA keys of all tokens when listing keys in the database.
----	--

---

---

<code>-d <i>keydir</i></code>	Specify a directory containing a key database file. On Unix Key Database Tool defaults to <code>\$HOME/.netscape</code> (that is, <code>~/netscape</code> ), and on Windows NT the default is the current directory.  The <code>key3.db</code> and <code>cert7.db</code> database files must reside in the same directory.
<code>-e <i>exp</i></code>	Set an alternate exponent value to use in generating a new RSA public key for the database, instead of the default value of 65537. The available alternate values are 3 and 17.
<code>-f <i>noise-file</i></code>	Read a seed value from the specified binary file to use in generating a new RSA private and public key pair. This argument makes it possible to use hardware-generated seed values and unnecessary to manually create a value from the keyboard. The minimum file size is 20 bytes.
<code>-h <i>tokenname</i></code>	Specify the name of a token to act on. Unless otherwise specified, the default token is an internal slot (specifically, internal slot 2). An internal slot is a virtual slot maintained in software, rather than a hardware device. Internal slot 2 is used by key and certificate services. Internal slot 1 is used by cryptographic services.  Use the Module Database Tool ( <code>modutil -list</code> ) to get a list of token names in the module database.
<code>-k <i>shortcutID</i></code>	Specify a private key by using the key identifier. You can use the complete keyID (as shown by the <code>-L</code> option), or the shortcutID. The shortcutID is the first few bytes of the keyID, starting from the second byte, with a length sufficient to identify it uniquely. If you specify a shortcutID that is not unique, the first private key that matches the shortcutID is found.
<code>-l</code>	List DSA as well as RSA keys when listing keys in the key database.
<code>-q <i>pqgfile</i></code>	Read an alternate PQG value from the specified file when generating DSA key pairs. If this argument is not used, Key Database Tool generates its own PQG value. PQG files are created with a separate DSA utility.

---



---

<code>-s <i>size</i></code>	Set a key size to use when generating new public and private key pairs. The minimum is 256 bits and the maximum is 1024 bits. The default is 1024 bits. Any size between the minimum and maximum is allowed.
<code>-t <i>rsa dsa</i></code>	Specify the type of a key, either RSA or DSA. The default value is <i>rsa</i> . By specifying the type of key you can avoid mistakes caused by duplicate shortkeyIDs.
<code>-w <i>password-file</i></code>	Specify a file to automatically supply the password necessary to access a key database. This is a plain-text file containing one password. You should not use this argument if you are accessing an internal slot and hardware tokens that use different passwords. Be sure to prevent unauthorized access to this file.

---

## Usage

Key Database Tool's capabilities are grouped as follows, using these combinations of options and arguments. The specifications in square brackets are optional, those without square brackets are required.

- Creating a new `key3.db` file and setting its password:

```
-N [-d keydir] [-w password-file]
```

- Changing the password to a key database file:

```
-C [-d keydir]
```

- Generating new RSA key pairs in a key database file:

```
-G [-h tokenname] [-t rsa] [-s num] [-e exp] [-d keydir]
[-f noise-file] [-w password-file]
```

- Generating new DSA key pairs in a key database file:

```
-G [-h tokenname] -t dsa [-q pqgfile -s num]
[-d keydir] [-w password-file]
```

- Listing the keyIDs of the keys in a database:

```
-L [-a] [-l] [-t rsa|dsa] [-h tokenname] [-d keydir]
```

- Displaying public key information from the database:

```
-P -k shortkeyID [-t rsa|dsa] [-h tokenname]  
[-d keydir] [-w password-file]
```

- Deleting private keys from a key database file:

```
-D -k shortkeyID [-t rsa|dsa] [-h tokenname]  
[-d keydir] [-w password-file]
```

- Displaying a list of the options and arguments used by Key Database Tool:

```
-H
```

## Examples

Includes the following:

- Creating a Key Database
- Generating a New Key
- Displaying Public Key Information
- Listing Key IDs
- Deleting a Private Key

## Creating a Key Database

This example creates new key database files (*key3.db* and *secmod.db*) in the specified directory:

```
keyutil -N -d keydir
```

Key Database Tool prompts you as follows:

```
Creating a brand new key database:keydir/key3.db  
Database not initialized. Setting password.  
Enter new password:
```

Re-enter password:

After you enter the password, Key Database Tool creates new `key3.db` and `secmod.db` files in the specified directory.

## Generating a New Key

This example generates a new key in a key database:

```
keyutil -G -d keydir
```

Key Database Tool then displays the following:

```
-----
Netscape Communications Corporation
Key Generation
-----

Welcome to the key generator. With this program, you can
generate the public and private keys that you use for secure
communications.

A random seed must be generated that will be used in the
creation of your key. One of the easiest ways to create a random
seed is to use the timing of keystrokes on a keyboard.

You have specified the name 'mykey' for your key

If this is correct, press enter:

To begin, type keys on the keyboard until this progress meter is
full. DO NOT USE THE AUTOREPEAT FUNCTION ON YOUR KEYBOARD!

Continue typing until the progress meter is full:

|*****|

Finished. Press enter to continue:

Generating key. This may take a few moments...

Password:

generated public/private key pair
```

Note that if you do not specify a token name, the key is generated on the internal slot. This is equivalent to the `-h internal` argument.

If you use the `-f noise-file` argument, Key Database Tool does not ask for keyboard input.

If you use the `-w password-file` argument, Key Database Tool reads the password from the file instead of asking for keyboard input. Avoid using this argument when you are accessing both the internal slot and tokens that have different passwords.

## Displaying Public Key Information

This example prints the public key's information:

```
keyutil -P -k e95c -d keydir
```

The public key information appears after you give the correct password:

```
Password:
```

```
It's the first key found.
```

```
RSA Public-Key:
```

```
modulus:
```

```
00:e9:5c:4a:73:74:39:22:6d:c6:da:4e:b3:1f:01:26:9d:be:
d1:74:ae:cd:c7:7d:65:f9:1d:31:1f:71:fb:60:d0:45:46:5f:
5a:19:e7:61:1e:e7:ce:9f:4a:13:4e:d6:e9:06:90:2a:ba:bd:
0b:5f:7b:a3:28:21:1e:0f:1c:f4:3a:ba:3a:8f:0b:e1:99:91:
cc:e8:fd:17:d2:1c:66:13:6b:95:27:b1:eb:bc:9c:e6:7b:f0:
3a:b9:44:dc:24:a6:f8:83:9a:9e:80:3f:74:48:09:6b:3f:a6:
46:51:be:e0:1b:51:87:8c:44:94:f0:fe:41:fe:b4:9f:4c:0a:
04:a9:a1
```

```
publicExponent: 65537 (0x10001)
```

## Listing Key IDs

This command lists the key IDs in the key database:

```
keyutil -L -d keydir
```

After you enter the password, Key Database Tool displays the following:

```
RSA Public-Key:
```

```
modulus:
```

```
00:e9:5c:4a:73:74:39:22:6d:c6:da:4e:b3:1f:01:26:9d:be:
d1:74:ae:cd:c7:7d:65:f9:1d:31:1f:71:fb:60:d0:45:46:5f:
```

```

5a:19:e7:61:1e:e7:ce:9f:4a:13:4e:d6:e9:06:90:2a:ba:bd:
0b:5f:7b:a3:28:21:1e:0f:1c:f4:3a:ba:3a:8f:0b:e1:99:91:
cc:e8:fd:17:d2:1c:66:13:6b:95:27:b1:eb:bc:9c:e6:7b:f0:
3a:b9:44:dc:24:a6:f8:83:9a:9e:80:3f:74:48:09:6b:3f:a6:
46:51:be:e0:1b:51:87:8c:44:94:f0:fe:41:fe:b4:9f:4c:0a:
04:a9:a1

```

When unmodified, this command lists all the RSA keys in the default (internal) slot. You can refine this command's output with the `-a`, `-h`, and `-l` arguments.

## Deleting a Private Key

This example deletes a private key from the key database:

```
keyutil -D -k e95c -d keydir
```

When you delete keys, be sure to remove any certificates associated with those keys from the certificate database by using the Certificate Database Tool.



## F

# Netscape Signing Tool

This appendix describes how to use version 1.2 of Netscape Signing Tool (`signtool` on the command line) to digitally sign software, including binary files intended for distribution via SmartUpdate, Java class files, and JavaScript scripts. Version 1.2 includes all the capabilities of, and is fully compatible with, previous versions of Netscape Signing Tool (.50, .60, 1.0, and 1.1).

This appendix has these sections:

- Introduction to Netscape Signing Tool (page 712)
- Using Netscape Signing Tool (page 716)
- SignTool Syntax and Options (page 722)
- Generating Test Object-Signing Certificates (page 733)
- Using Netscape Signing Tool with Smart Cards (page 735)
- Netscape Signing Tool and FIPS-140-1 (page 738)
- Answers to Common Questions (page 740)

# Introduction to Netscape Signing Tool

This section reviews basic concepts that you need to understand before you begin using version 1.2 of Netscape Signing Tool to sign files or JavaScript scripts. If you are already familiar with object-signing concepts, go straight to “Using Netscape Signing Tool” on page 716.

- What Is Netscape Signing Tool?
- JAR Format and JAR Archives
- What Signing a File Means
- Object-Signing Certificates

For a complete introduction to object signing technology, see *Netscape Object Signing: Establishing Trust for Downloaded Software* at this URL:

`http://developer.netscape.com/docs/manuals/signedobj/trust/index.htm`

## What Is Netscape Signing Tool?

Netscape Signing Tool is a stand-alone command-line tool that creates digital signatures and uses the Java Archive (JAR) format to associate them with files in a directory. It is intended for use by system administrators and by developers who want to distribute software electronically over the Internet.

Netscape Signing Tool 1.2 is available for the following platforms:

- Windows NT 4.0
- Solaris 2.5.1
- AIX 4.2
- HP-UX 11.0
- Linux ELF glibc



This appendix describes how to use Netscape Signing Tool 1.2 to sign Java applets, JavaScript scripts, plug-ins, and other files and how to package the signed objects in a *JAR archive* (also called a *JAR file*), which is a digital envelope for a compressed collection of files. Communicator client software uses JAR archives to install or update software automatically.

Electronic software distribution over any network involves potential security problems. To help address some of these problems, you can associate digital signatures with the files in a JAR archive. Digital signatures allow Communicator to perform two operations that are important to end users:

- Confirm the identity of the individual, company, or other entity whose digital signature is associated with the files
- Check whether the files have been tampered with since being signed

You do not need to understand the technical details of JAR archives or digital signatures to use Netscape Signing Tool. However, you do need some familiarity with the concepts described in the rest of this appendix. If you are already familiar with basic object-signing concepts, go straight to “Using Netscape Signing Tool” on page 716.

## JAR Format and JAR Archives

The *Java Archive (JAR) format* **is** a set of conventions for associating digital signatures, installer scripts, and other information with files in a directory. Signing tools such as Netscape Signing Tool allow you to sign files using the JAR format and package them as a single JAR file. JAR files are used by Communicator client software to support automatic software installation, user-controlled access to local system resources by Java applets, and other features that help address potential security problems.

The *JAR file type* is a registered Internet MIME type based on the standard cross-platform ZIP archive format. A JAR file functions as a digital envelope for a compressed collection of files. The JAR file type is distinct from the JAR format, which is simply a way of organizing information in a directory.

Because the JAR format doesn’t require a digital signature to be stored physically inside the file with which it is associated, JAR files are extremely flexible. You can use Netscape Signing Tool to sign any files, including Java

class files, Communicator plug-ins, or any other kind of document or application. You can also use version 1.1 and later versions of Netscape Signing Tool to sign inline JavaScript scripts.

You must create a JAR file if you want to take advantage of Communicator's SmartUpdate feature. Communicator can automatically locate, download, and install components, plug-ins, and Java classes on a user's machine, thus freeing the user from this chore. Automatic software installation also helps both software developers who want to distribute software and updates over the Internet and system administrators using Mission Control to manage a corporate intranet.

You don't need to know anything else about the JAR format to use Netscape Signing Tool, which takes care of the details for you. For detailed information about the JAR format, see *The Jar Format* at this URL:

<http://developer.netscape.com/docs/manuals/signedobj/jarfile/index.html>

For detailed information about using the JAR Installation Manager to package your software for use with SmartUpdate, see *Using JAR Installation Manager for SmartUpdate* at this URL:

<http://developer.netscape.com/docs/manuals/communicator/jarman/index.htm>

## What Signing a File Means

If you have a signing certificate, you can use Netscape Signing Tool to digitally sign files and package them as a JAR file. An *object-signing certificate* is a special kind of certificate that allows you to associate your digital signature with one or more files. For information about obtaining an object-signing certificate, see *Object-Signing Tools* at this URL:

<http://developer.netscape.com/docs/manuals/index.html?content=security.html>

An individual file can potentially be signed with multiple digital signatures. For example, a commercial software developer might sign the files that constitute a software product to prove that the files are indeed from a particular company. A network administrator might sign the same files with an additional digital signature based on a company-generated certificate to indicate that the product is approved for use within the company.

The significance of a digital signature is comparable to the significance of a handwritten signature. Once you have signed a file, it is difficult to claim later that you didn't sign it. In some situations, a digital signature may be considered as legally binding as a handwritten signature. Therefore, you should take great care to ensure that you can stand behind any file you sign and distribute.

For example, if you are a software developer, you should test your code to make sure it is virus-free before signing it. Similarly, if you are a network administrator, you should make sure, before signing any code, that it comes from a reliable source and will run correctly with the software installed on the machines to which you are distributing it.

## Object-Signing Certificates

Before you can use Netscape Signing Tool to sign files, you must have an object-signing certificate, which is a special certificate whose associated private key is used to create digital signatures.

For testing purposes only, you can create an object-signing certificate with Netscape Signing Tool 1.2. When testing is finished and you are ready to distribute your software, you should obtain an object-signing certificate from one of two kinds of sources:

- An independent certificate authority (CA) that authenticates your identity and charges you a fee. You typically get a certificate from an independent CA if you want to sign software that will be distributed over the Internet.
- CA server software running on your corporate intranet or extranet. Netscape Certificate Management System provides a complete management solution for creating, deploying, and managing certificates, including CAs that issue object-signing certificates.

You must also have a certificate for the CA that issues your signing certificate before you can sign files. If the certificate authority's certificate isn't already installed in your copy of Communicator, you typically install it by clicking the appropriate link on the certificate authority's web site, for example on the page from which you initiated enrollment for your signing certificate. This is the case for some test certificates, as well as certificates issued by Netscape Certificate Management System: you must download the the CA certificate in addition to obtaining your own signing certificate. CA certificates for several certificate authorities are preinstalled in the Communicator certificate database.

When you receive an object-signing certificate for your own use, it is automatically installed in your copy of the Communicator client software. Communicator supports the public-key cryptography standard known as PKCS #12, which governs key portability. You can, for example, move an object-signing certificate and its associated private key from one computer to another on a credit-card-sized device called a smart card. For more information, see “Using Netscape Signing Tool with Smart Cards” on page 735.

## Using Netscape Signing Tool

This section describes how to use Netscape Signing Tool to create digital signatures for files in a directory and to associate the signatures with the files according to the JAR format. Netscape Signing Tool also provides an option that automatically creates a JAR file containing the directory; this option was not implemented in pre-1.0 versions. For maximum flexibility, and for compatibility with scripts that used earlier versions of Netscape Signing Tool, you can still use a ZIP utility to create the JAR file.

- Getting Ready to Use Netscape Signing Tool
- Signing a File
- Using Netscape Signing Tool with a ZIP Utility
- Tips and Techniques

For a complete list of Netscape Signing Tool command-line options, see “SignTool Syntax and Options” on page 722.

## Getting Ready to Use Netscape Signing Tool

Before using Netscape Signing Tool, you must have the `signtool` executable in your path environment variable. You must also have an object-signing certificate.

Netscape Signing Tool includes an option that allows you to generate an object-signing certificate for testing purposes. For information about using this option, see “Generating Test Object-Signing Certificates” on page 733.

Although suitable for testing purposes, the object-signing certificate produced by Netscape Signing Tool is not recommended for signing finished software that will be widely distributed over the Internet or an intranet. When you are ready to sign finished software, you will need to get an object-signing certificate from your company's internal certificate authority, if it has one, or from a third-party certificate authority.

The sections that follow describe how to prepare Netscape Signing Tool for signing files.

## Setting Up Your Certificate

These instructions apply to an object-signing certificate obtained from a third party or an in-house CA for use in signing finished code. During development, you may wish to use a special certificate generated by Netscape Signing Tool for testing purposes.

If you obtained your object-signing certificate while running Communicator on a system that's different from the system on which you intend to sign files, you need to copy your certificate and private key files to the new system. Communicator's certificate and key databases are portable among all platforms.

On the computer where you ran Communicator to get the object-signing certificate, locate the files `key3.db` and `cert7.db`. For example, on a typical Windows NT system, these files are found at `C:\Program Files\NETSCAPE\USERS\username\`. You must copy these files to the system where you intend to sign pages. (If you use FTP, be sure to transfer in binary mode.)

If you are running Netscape Signing Tool on a Unix system and you don't already have a `~/netscape` directory, first run Communicator once to create one. If you want to maintain whatever certificates are already in your `~/netscape` directory, put the existing `key3.db` and `cert7.db` files in some other directory before replacing them with the versions that include the object-signing certificate you want to use with Netscape Signing Tool.

If you are using Unix, set up an alias to call `signtool`, or place it in your path.

If you are using Windows 95 or NT, the `signtool` executable doesn't know where your certificates are, so either put the `key3.db` and `cert7.db` files in the current directory and use `-d.` or use `-d` to point to the directory in which they are located.

**Warning** Keep copies of the key3.db and cert7.db files somewhere separate from the copies you use with the signtool executable. This ensures that you won't lose your certificates if you accidentally damage the files. If your keys are on external tokens, such as smart cards, you should keep a copy the secmod.db file.

## Listing Available Certificates

You use the -L option to list the nicknames for all available certificates and check which ones are signing certificates, as shown in this Unix example:

```
% signtool -L
using certificate directory: /u/jsmith/.netscape
S Certificates
- -----
  BBN Certificate Services CA Root 1
  IBM World Registry CA
  VeriSign Class 1 CA - Individual Subscriber - VeriSign, Inc.
  GTE CyberTrust Root CA
  Uptime Group Plc. Class 4 CA
* Verisign Object Signing Cert
  Integrion CA
  GTE CyberTrust Secure Server CA
  AT&T Directory Services
* test object signing cert
  Uptime Group Plc. Class 1 CA
  VeriSign Class 1 Primary CA
- -----
Certificates that can be used to sign objects have *'s to their left.
%
```

In the above example, two signing certificates are displayed: Verisign Object Signing Cert and test object signing cert.

You use the -l option to get a list of signing certificates only, including the signing CA for each, as shown in this Unix example:

```
% signtool -l
using certificate directory: /u/jsmith/.netscape
Object signing certificates
-----
Verisign Object Signing Cert
    Issued by: VeriSign, Inc. - Verisign, Inc.
    Expires: Tue May 19, 1998
test object signing cert
    Issued by: test object signing cert (Signtool 1.0 Testing
Certificate (960187691))
    Expires: Sun May 17, 1998
```

-----  
 For a list including CAs, use "signtool -L"

## Signing a File

To sign a file using Netscape Signing Tool, follow these steps:

### 1. Create an empty directory.

```
% mkdir signdir
```

### 2. Put some file into it.

```
% echo boo > signdir/test.f
```

### 3. Specify the name of your object-signing certificate and sign the directory.

If you are using Unix, this example assumes you have put your .db files in the

~/.netscape directory, as explained in "Setting Up Your Certificate" on page 717.

```
% signtool -k MySignCert -Z testjar.jar signdir
```

```
using key "MySignCert"
using certificate directory: /u/jsmith/.netscape
Generating signdir/META-INF/manifest.mf file..
--> test.f
adding signdir/test.f to testjar.jar
Generating signtool.sf file..
Enter Password or Pin for "Communicator Certificate DB":
```

### 4. At the prompt, type the password to your private-key database.

If it accepts the password, signtool responds as follows:

```
adding signdir/META-INF/manifest.mf to testjar.jar
adding signdir/META-INF/signtool.sf to testjar.jar
adding signdir/META-INF/signtool.rsa to testjar.jar
tree "signdir" signed successfully
```

### 5. Test the archive you just created.

```
% signtool -v testjar.jar
```

```
using certificate directory: /u/jsmith/.netscape
archive "testjar.jar" has passed crypto verification.
```

status	path
verified	test.f

You can also use Netscape Signing Tool from within a script to automate some aspects of signing. For example, here's a Windows script that starts with an unsigned JAR file, unpackages it, signs it, and then repackages it:

```
rem Expand the jar file into a new directory
unzip -qq myjar.jar -d signjar
del myjar.jar
rem Sign everything in the new directory and recompress
signtool -k MySignCert -Z myjar.jar signdir
```

## Using Netscape Signing Tool with a ZIP Utility

To use Netscape Signing Tool with a ZIP utility, you must have the utility in your path environment variable. You should use the `zip.exe` utility rather than `pkzip.exe`, which cannot handle long filenames.

You can use a ZIP utility instead of the `-Z` option to package a signed archive into a JAR file after you have signed it:

```
% cd signdir
% zip -r ../myjar.jar *
  adding: META-INF/ (stored 0%)
  adding: META-INF/manifest.mf (deflated 15%)
  adding: META-INF/signtool.sf (deflated 28%)
  adding: META-INF/signtool.rsa (stored 0%)
  adding: text.txt (stored 0%)
%
```

## Tips and Techniques

- If you are storing JAR files or their components in CVS, store them as binary files.
- If you are signing metadata only and not files, you still need to create a blank directory for Netscape Signing Tool to sign.



- When using the Windows NT version of Netscape Signing Tool, always use a relative path.
- The command `signtool -L` should list your object-signing certificate with an asterisk (\*) beside it. If it doesn't, you cannot sign files.
- If you are having problems using the JAR file from within Navigator, check `signtool -v` on your final archive.
- Don't use Netscape Signing Tool's `-G` option while Communicator is running. The `-G` option writes to the security databases as it generates certificates, and corruption could occur if Communicator simultaneously attempts to write to these files. All other Netscape Signing Tool options are read-only and can't harm these files.
- If you see the error `Unknown issuer`, you need to get the certificate of the certificate authority that issued your signing certificate. Alternatively, you may have the certificate authority's certificate, but it may not be trusted for object signing.
- If you see the error `Issuer not trusted`, open Communicator on the system you used when you obtained the certificate and follow these steps:
  1. Click the Security button in a Navigator window.
  2. Click Signers under Certificates in the left frame.
  3. Select the CA that issued your signing certificate.
  4. Click the Edit button.
  5. Select the "Accept this Certificate Authority for Certifying software developers" checkbox.
  6. Click OK.

You then need to transfer the `cert7.db` file to the appropriate directory on the system on which you are running Netscape Signing Tool, as described in "Setting Up Your Certificate" on page 717.

# SignTool Syntax and Options

This section summarizes the syntax and options for Netscape Signing Tool 1.2.

- Command Syntax
- Command Options
- Command File Syntax
- Command File Keywords and Example

## Command Syntax

To run Netscape Signing Tool, type

```
signtool options
```

where *options* can be any sequence of the options listed in this chapter.

Each argument for each `signtool` option must be in quotes if it contains any spaces or other nonalphanumeric characters.

## Command Options

Options for `signtool` are defined as follows:

---

<code>-b <i>basename</i></code>	<p>Specifies the base filename for the <code>.rsa</code> and <code>.sf</code> files in the META-INF directory (required by The JAR Format). For example,</p> <p><code>-b signatures</code></p> <p>causes the files to be named <code>signatures.rsa</code> and <code>signatures.sf</code>. The default is <code>signtool</code>.</p> <p>The <code>-b</code> option is available in Netscape Signing Tool 1.0 and later versions only.</p>
<code>-c#</code>	<p>Specifies the compression level for the <code>-J</code> or <code>-Z</code> option. The symbol <code>#</code> represents a number from 0 to 9, where 0 means no compression and 9 means maximum compression. The higher the level of compression, the smaller the output but the longer the operation takes.</p> <p>If the <code>-c#</code> option is not used with either the <code>-J</code> or the <code>-Z</code> option, the default compression value used by both the <code>-J</code> and <code>-Z</code> options is 6.</p>
<code>-d <i>certdir</i></code>	<p>Specifies your certificate database directory; that is, the directory in which you placed your <code>key3.db</code> and <code>cert7.db</code> files. To specify the current directory, use <code>“-d.”</code> (including the period).</p> <p>The Unix version of <code>signtool</code> assumes <code>~/netscape</code> unless told otherwise. The NT version of <code>signtool</code> always requires the use of the <code>-d</code> option to specify where the database files are located.</p>

---

---

<code>-e <i>extension</i></code>	Tells <code>signtool</code> to sign only files with the given extension; for example, use <code>-e ".class"</code> to sign only Java class files. Note that with Netscape Signing Tool version 1.1 and later this option can appear multiple times on one command line, making it possible to specify multiple file types or classes to include.
<code>-f <i>commandfile</i></code>	Specifies a text file containing Netscape Signing Tool options and arguments in <i>keyword=value</i> format. All options and arguments can be expressed through this file. For more information about the syntax used with this file, see “Tips and Techniques” on page 720.

---

---

***-G nickname***

Generates a new private-public key pair and corresponding object-signing certificate with the given nickname.

The newly generated keys and certificate are installed into the key and certificate databases in the directory specified by the **-d** option. With the NT version of Netscape Signing Tool, you must use the **-d** option with the **-G** option. With the Unix version of Netscape Signing Tool, omitting the **-d** option causes the tool to install the keys and certificate in the Communicator key and certificate databases. If you are installing the keys and certificate in the Communicator databases, you must exit Communicator before using this option; otherwise, you risk corrupting the databases. In all cases, the certificate is also output to a file named `x509.cacert`, which has the MIME-type `application/x-x509-ca-cert`.

Unlike certificates normally used to sign finished code to be distributed over a network, a test certificate created with **-G** is not signed by a recognized certificate authority. Instead, it is self-signed. In addition, a single test signing certificate functions as both an object-signing certificate and a CA. When you are using it to sign objects, it behaves like an object-signing certificate. When it is imported into browser software such as Communicator, it behaves like an object-signing CA and cannot be used to sign objects.

The **-G** option is available in Netscape Signing Tool 1.0 and later versions only. By default, it produces only RSA certificates with 1024-byte keys in the internal token. However, you can use the **-s** option specify the required key size and the **-t** option to specify the token. For more information about the use of the **-G** option, see “Generating Test Object-Signing Certificates” “Generating Test Object-Signing Certificates” on page 733.

---

---

<code>-i <i>scriptname</i></code>	Specifies the name of an installer script for SmartUpdate. This script installs files from the JAR archive in the local system after SmartUpdate has validated the digital signature. For more details, see the description of <code>-m</code> that follows. The <code>-i</code> option provides a straightforward way to provide this information if you don't need to specify any metadata other than an installer script.
<code>-j <i>directory</i></code>	Specifies a special JavaScript directory. This option causes the specified directory to be signed and tags its entries as inline JavaScript. This special type of entry does not have to appear in the JAR file itself. Instead, it is located in the HTML page containing the inline scripts. When you use <code>signtool -v</code> , these entries are displayed with the string <code>NOT PRESENT</code> .
<code>-J</code>	<p>Signs a directory of HTML files containing JavaScript and creates as many archive files as are specified in the HTML tags. Even if <code>signtool</code> creates more than one archive file, you need to supply the key database password only once.</p> <p>The <code>-J</code> option is available only in Netscape Signing Tool 1.0 and later versions. The <code>-J</code> option cannot be used at the same time as the <code>-Z</code> option.</p> <p>If the <code>-c#</code> option is not used with the <code>-J</code> option, the default compression value is 6.</p> <p>Note that versions 1.1 and later of Netscape Signing Tool correctly recognizes the <code>CODEBASE</code> attribute, allows paths to be expressed for the <code>CLASS</code> and <code>SRC</code> attributes instead of filenames only, processes <code>LINK</code> tags and parses HTML correctly, and offers clearer error messages.</p>

---

---

<code>-k <i>key</i> . . . <i>directory</i></code>	<p>Specifies the nickname (<i>key</i>) of the certificate you want to sign with and signs the files in the specified directory. The directory to sign is always specified as the last command-line argument. Thus, it is possible to write</p> <pre>signtool -k MyCert -d . signdir</pre> <p>You may have trouble if the nickname contains a single quotation mark. To avoid problems, escape the quotation mark using the escape conventions for your platform.</p> <p>It's also possible to use the <code>-k</code> option without signing any files or specifying a directory. For example, you can use it with the <code>-l</code> option to get detailed information about a particular signing certificate.</p>
<code>-l</code>	<p>Lists signing certificates, including issuing CAs. If any of your certificates are expired or invalid, the list will so specify. This option can be used with the <code>-k</code> option to list detailed information about a particular signing certificate.</p> <p>The <code>-l</code> option is available in Netscape Signing Tool 1.0 and later versions only.</p>
<code>-L</code>	<p>Lists the certificates in your database. An asterisk appears to the left of the nickname for any certificate that can be used to sign objects with <code>signtool</code>.</p>
<code>--leavearc</code>	<p>Retains the temporary <code>.arc</code> (archive) directories that the <code>-J</code> option creates. These directories are automatically erased by default. Retaining the temporary directories can be an aid to debugging.</p>

---

---

`-m metafile`

Specifies the name of a metadata control file. Metadata is signed information attached either to the JAR archive itself or to files within the archive. This metadata can be any ASCII string, but is used mainly for specifying an installer script.

The metadata file contains one entry per line, each with three fields:

field #1: file specification, or + if you want to specify global metadata (that is, metadata about the JAR archive itself or all entries in the archive)

field #2: the name of the data you are specifying; for example: `Install-Script`

field #3: data corresponding to the name in field #2

For example, the `-i` option uses the equivalent of this line:

```
+ Install-Script: script.js
```

This example associates a MIME type with a file:

```
movie.qt MIME-Type: video/quicktime
```

For information about the way installer script information appears in the manifest file for a JAR archive, see [The JAR Format](#) on Netscape DevEdge.

`-M`

Lists the PKCS #11 modules available to `signtool`, including smart cards.

The `-M` option is available in Netscape Signing Tool 1.0 and later versions only.

For information on using Netscape Signing Tool with smart cards, see “Using Netscape Signing Tool with Smart Cards” on page 735.

For information on using the `-M` option to verify FIPS-140-1 validated mode, see “Netscape Signing Tool and FIPS-140-1” on page 738.

---



---

<code>--norecurse</code>	Blocks recursion into subdirectories when signing a directory's contents or when parsing HTML.
<code>-o</code>	Optimizes the archive for size. Use this <i>only</i> if you are signing very large archives containing hundreds of files. This option makes the manifest files (required by the JAR format) considerably smaller, but they contain slightly less information.
<code>--outfile <i>outputfile</i></code>	Specifies a file to receive redirected output from Netscape Signing Tool.
<code>-p <i>password</i></code>	Specifies a password for the private-key database. Note that the password entered on the command line is displayed as plain text.
<code>-s <i>keysize</i></code>	<p>Specifies the size of the key for generated certificate. Use the <code>-M</code> option to find out what tokens are available.</p> <p>The <code>-s</code> option can be used with the <code>-G</code> option only.</p>
<code>-t <i>token</i></code>	<p>Specifies which available token should generate the key and receive the certificate. Use the <code>-M</code> option to find out what tokens are available.</p> <p>The <code>-t</code> option can be used with the <code>-G</code> option only.</p>
<code>-v <i>archive</i></code>	Displays the contents of an archive and verifies the cryptographic integrity of the digital signatures it contains and the files with which they are associated. This includes checking that the certificate for the issuer of the object-signing certificate is listed in the certificate database, that the CA's digital signature on the object-signing certificate is valid, that the relevant certificates have not expired, and so on.

---

---

<code>--verbosity <i>value</i></code>	Sets the quantity of information Netscape Signing Tool generates in operation. A value of 0 (zero) is the default and gives full information. A value of -1 suppresses most messages, but not error messages.
<code>-w <i>archive</i></code>	Displays the names of signers of any files in the archive.
<code>-x <i>directory</i></code>	Excludes the specified directory from signing. Note that with Netscape Signing Tool version 1.1 and later this option can appear multiple times on one command line, making it possible to specify several particular directories to exclude.
<code>-z</code>	Tells <code>signtool</code> not to store the signing time in the digital signature. This option is useful if you want the expiration date of the signature checked against the current date and time rather than the time the files were signed.
<code>-Z <i>jarfile</i></code>	Creates a JAR file with the specified name. You must specify this option if you want <code>signtool</code> to create the JAR file; it does not do so automatically. If you don't specify -Z, you must use an external ZIP tool to create the JAR file.
	The -Z option cannot be used at the same time as the -J option.
	If the -c# option is not used with the -Z option, the default compression value is 6.

---

## Command File Syntax

Entries in a Netscape Signing Tool command file have this general format:

*keyword=value*

Everything before the = sign on a single line is a keyword, and everything from the = sign to the end of line is a value. The value may include = signs; only the first = sign on a line is interpreted. Blank lines are ignored, but white space on a line with keywords and values is assumed to be part of the keyword (if it comes before the equal sign) or part of the value (if it comes after the first equal sign). Keywords are case insensitive, values are generally case sensitive. Since the = sign and newline delimit the value, it should not be quoted.

## Command File Keywords and Example

Here are the command file keywords and their values:

Keyword	Value
basename	Same as -b option.
compression	Same as -c option.
certdir	Same as -d option.
extension	Same as -e option.
generate	Same as -G option.
installscript	Same as -i option.
javascriptdir	Same as -j option.
htmldir	Same as -J option.
certname	Nickname of certificate, as with -k and -l -k options.
signdir	The directory to be signed, as with -k option.
list	Same as -l option. Value is ignored, but = sign must be present.

Keyword	Value
<code>listall</code>	Same as <code>-L</code> option. Value is ignored, but <code>=</code> sign must be present.
<code>metafile</code>	Same as <code>-m</code> option.
<code>modules</code>	Same as <code>-M</code> option. Value is ignored, but <code>=</code> sign must be present.
<code>optimize</code>	Same as <code>-o</code> option. Value is ignored, but <code>=</code> sign must be present.
<code>password</code>	Same as <code>-p</code> option.
<code>keysize</code>	Same as <code>-s</code> option.
<code>token</code>	Same as <code>-t</code> option.
<code>verify</code>	Same as <code>-v</code> option.
<code>who</code>	Same as <code>-w</code> option.
<code>exclude</code>	Same as <code>-x</code> option.
<code>notime</code>	Same as <code>-z</code> option. value is ignored, but <code>=</code> sign must be present.
<code>jarfile</code>	Same as <code>-Z</code> option.
<code>outfile</code>	Name of a file to which output and error messages will be redirected. This option has no command-line equivalent.

Here's an example of the use of the command file. The command

```
signtool -d c:\netscape\users\james -k mycert -Z myjar.jar signdir > output.txt
```

becomes

```
signtool -f somefile
```

where *somefile* contains the following lines:

```
certdir=c:\netscape\users\james
certname=mycert
jarfile=myjar.jar
signdir=signdir
outfile=output.txt
```

## Generating Test Object-Signing Certificates

Netscape Signing Tool allows you to create object-signing certificates for testing purposes. This section describes how to create and use such test certificates.

Unlike certificates normally used to sign finished code to be distributed over a network, the test certificates created with Netscape Signing Tool are not signed by a recognized certificate authority. Instead, they are self-signed. In addition, a single test signing certificate functions as both an object-signing certificate and a CA. When you are using it to sign objects, it behaves like an object-signing certificate. When it is imported into browser software such as Communicator, it behaves like an object-signing CA.

## Generating the Keys and Certificate

The `signtool` option `-G` generates a new public-private key pair and certificate. It takes the nickname of the new certificate as an argument. The newly generated keys and certificate are installed into the key and certificate databases in the directory specified by the `-d` option. With the NT version of Netscape Signing Tool, you must use the `-d` option with the `-G` option. With the Unix version of Netscape Signing Tool, omitting the `-d` option causes the tool to install the keys and certificate in the Communicator key and certificate databases. In all cases, the certificate is also output to a file named `x509.cacert`, which has the MIME-type `application/x-x509-ca-cert`.

**Important** Before installing new keys and certificates in the key and certificate databases, you must set the database password (if you have not done so already). To set the password for the key and certificate databases currently being used by Communicator, click the Security icon in the Communicator toolbar, click Passwords, and click Set Password to create a password.

**Warning** If you intend to install the new key pair and certificate in the Communicator databases, you must exit Communicator before using Netscape Signing Tool to generate the object-signing certificate. Otherwise, you risk corrupting your certificate and key databases.

Certificates contain standard information about the entity they identify, such as the common name and organization name. Netscape Signing Tool prompts you for this information when you run the command with the `-G` option. However, all of the requested fields are optional for test certificates. If you do not enter a common name, the tool provides a default name. In the following example, the user input is in boldface:

```
% signtool -G MyTestCert
using certificate directory: /u/someuser/.netscape
Enter certificate information. All fields are optional. Acceptable
characters are numbers, letters, spaces, and apostrophes.
certificate common name: Test Object Signing Certificate
organization: Netscape Communications Corp.
organization unit: Server Products Division
state or province: California
country (must be exactly 2 characters): US
username: someuser
email address: someuser@netscape.com
Enter Password or Pin for "Communicator Certificate DB": [Password will
not echo]
generated public/private key pair
certificate request generated
certificate has been signed
certificate "MyTestCert" added to database
Exported certificate to x509.raw and x509.cacert.
%
```

The certificate information is read from standard input. Therefore, the information can be read from a file using the redirection operator (`<`) in some operating systems. To create a file for this purpose, enter each of the seven input fields, in order, on a separate line. Make sure there is a newline character at the end of the last line. Then run `signtool` with standard input redirected from your file as follows:

```
% signtool -G MyTestCert <inputfile
```

The prompts show up on the screen, but the responses will be automatically read from the file. The password will still be read from the console unless you use the `-p` option to give the password on the command line.

## Using Netscape Signing Tool with Smart Cards

This section describes how to use smart cards from within Netscape Signing Tool to digitally sign files.

- What Is a Smart Card?
- Setting Up a Smart Card
- Using the `-M` Option to List Smart Cards
- Using Netscape Signing Tool and a Smart Card to Sign Files

### What Is a Smart Card?

A *smart card* (sometimes called a *token*) is a credit-card-sized card, a key, or other easily removable device that can be used for cryptographic operations and for storing certificates. Smart cards are portable and must be physically inserted in an appropriate smart card reader attached to a computer for use with Communicator software running on that computer. Smart cards extend the private-key protection provided by Communicator, since private keys stored on the card require the card's presence as well as the password to the private-key database.

Navigator and Netscape Signing Tool support PKCS #11, a cryptographic standard developed to support services provided by smart cards. Before purchasing a smart card for use with Communicator, you should ensure that your vendor provides a PKCS #11 driver that has been tested with Communicator on your platform. Tested brands include [Litronic Netsign](#) and [Datakey's SignaSURE](#).

## Setting Up a Smart Card

Connect the smart card reader according to the manufacturer instructions. You may need to reset the smart card to a default state using the manufacturer's configuration utility. Not all smart cards require this step.

Smart cards designed for use with Communicator come with a software driver that you should install in your computer according to the manufacturer's instructions. You can then add the driver (also called a *cryptographic module*) to Communicator as follows:

1. Make sure the smart card is inserted in the smart card reader.
2. Click the Security button near the top of a Navigator window.
3. Click Cryptographic Modules in the left frame.
4. Click the Add button.
5. Type an appropriate name for the module you want to add in the box labeled Security Module Name.
6. Type the name of the driver that was supplied with your smart card in the box labeled Security Module File. For Windows systems, this is a dynamic linked library (DLL). You don't have to type the entire path, but you may.
7. Click OK.
8. If Communicator asks for it, type the smart card password.
9. Select the module you've just installed and click the View/Edit button.
10. Make sure the displayed information is correct for the smart card you just installed.
11. Select the name of the smart card.
12. Click the More Info button and examine that information as well.



**13.** If the state of the smart card (shown near the bottom of the More Info window) is Not Logged In, click OK and then click the Login button. Otherwise, just click OK. (Logging in allows you to install your signing certificate on the smart card. The smart card doesn't have to be logged in within Communicator for you to use it with Netscape Signing Tool.)

**14.** Click OK again.

After you have activated the smart card, use Communicator to visit the web site for the certificate authority (CA) you want to use and request a signing certificate.

When you submit your information to the certificate authority, Communicator asks you to select the card or database you wish to use to generate your private key. You should select the name of your smart card.

Your system then generates a public-private key pair and submits your request to the CA. When you receive the certificate, it is installed directly onto the card and travels with that smart card. However, you will be unable to use the certificate unless the smart card is inserted in the appropriate reader and you have entered its password correctly.

## Using the -M Option to List Smart Cards

You can use the -M option to list the PKCS #11 modules, including smart cards, that are available to signtool:

```
% signtool -d "c:\netscape\users\jsmith" -M
using certificate directory: c:\netscape\users\

```

```

        slot: Litronic 210
        token:
3. Datakey SignaSURE
        (this is an external module)
DLL name: dkck232.dll
        slots: 1 slots attached
        status: loaded
        slot: Datakey Reader
        token: <username>
-----

```

## Using Netscape Signing Tool and a Smart Card to Sign Files

Before you try to use Netscape Signing Tool with a smart card, try using it to sign a file without a smart card as described in “Using Netscape Signing Tool” on page 716.

The `signtool` command normally takes an argument of the `-k` option to specify a signing certificate. To sign with a smart card, you supply only the fully qualified name of the certificate.

To see fully qualified certificate names when you run Communicator, click the Security button in Navigator, then click Yours under Certificates in the left frame. Fully qualified names are of the format *smart card:certificate*, for example "MyCard:My Signing Cert". You use this name with the `-k` argument as follows:

```
signtool -k "MyCard:My Signing Cert" directory
```

where *directory* is the directory tree you want to sign. `signtool` asks you for two passwords: the password that protects the Communicator certificate database and the password that protects your smart card. If the passwords are correct, `signtool` signs the files in the directory.

## Netscape Signing Tool and FIPS-140-1

This section describes how to use Netscape Signing Tool in FIPS-140-1 validated mode. FIPS 140-1 is a U.S. government standard for implementations of cryptographic modules—that is, hardware or software that encrypts and

decrypts data or performs other cryptographic operations (such as creating or verifying digital signatures). Many products sold to the U.S. government must comply with one or more of the FIPS standards.

- Using FIPS-140 Mode
- Verifying FIPS Mode

For general information on FIPS standards and Netscape FIPS-140-1 validation, see the FIPS 140-1 FAQ.

## Using FIPS-140 Mode

Netscape Signing Tool is FIPS-140-1 validated when it uses the FIPS-validated Netscape cryptographic module. The FIPS module can be activated and deactivated from within Communicator. Communicator stores the module choice in the security module database (called `secmod.db` on Windows platforms and `secmodule.db` on Unix platforms). This database is stored in the same directory as your certificate database (`cert7.db`) and key database (`key3.db`), as indicated by the `-d` option of Netscape Signing Tool.

Before using Netscape Signing Tool in FIPS-validated mode, you must use Navigator to switch to FIPS mode. For information on how to do this, see Operating Netscape Navigator in FIPS PUB-140-1 Compliant Mode on Netscape DevEdge.

After switching the Navigator cryptographic module to FIPS mode, you have two choices:

- Use the same security module database from Netscape Signing Tool (by specifying the same directory with the `-d` option).
- Make a copy of Communicator's security module database and place it in Netscape Signing Tool's database directory.

## Verifying FIPS Mode

Use the `-M` option to verify that you are using the FIPS-140-1 module.

This Unix example shows that Netscape Signing Tool is using a non-FIPS module:

```
% signtool -d "c:\netscape\users\jsmith" -M
using certificate directory: c:\netscape\users\jsmith
Listing of PKCS11 modules
-----
1. Netscape Internal PKCS #11 Module
   (this module is internally loaded)
   slots: 2 slots attached
   status: loaded
   slot: Communicator Internal Cryptographic Services Version 4.0
   token: Communicator Generic Crypto Svcs
   slot: Communicator User Private Key and Certificate Services
   token: Communicator Certificate DB
-----
```

This Unix example shows that Netscape Signing Tool is using a FIPS-140-1 module:

```
% signtool -d "c:\netscape\users\jsmith" -M
using certificate directory: c:\netscape\users\jsmith
Enter Password or Pin for "Communicator Certificate DB": [password will not
echo]
Listing of PKCS11 modules
-----
1. Netscape Internal FIPS PKCS #11 Module
   (this module is internally loaded)
   slots: 1 slots attached
   status: loaded
   slot: Netscape Internal FIPS-140-1 Cryptographic Services
   token: Communicator Certificate DB
-----
```

## Answers to Common Questions

This section answers the most common technical questions regarding Netscape Signing Tool.

**Netscape Signing Tool, or Communicator, fails to report the presence of a particular certificate in the database, even though that certificate should be there.**

Netscape Signing Tool 1.x and Communicator 4.x are designed to read only the `cert7.db` files used by Communicator 4.x. If it happens that a certificate gets downloaded with Navigator 3.x *after* Communicator 4.x was installed and run, that certificate is recorded in a database of the older (3.x) format. While Communicator does automatically convert Navigator's `cert5.db` and `key.db` databases to the `cert7.db` and `key3.db` formats the first time it runs, it does not do so again.

To get a certificate into the new database from an old one requires forcing Communicator to reinitialize its `cert7.db` file as it does at first run-time. This requires that the certificates in the current `cert7.db` file be exported for later re-importing.

1. Click the Security Info button on the Communicator toolbar.
2. Click Yours under Certificates in the left panel, and select a certificate to export.
3. Click Export and save a PKCS #12 copy of the certificate to a safe location (if no copy already exists).
4. Repeat steps 2 and 3 for each certificate present.
5. Exit Communicator completely.
6. Move the `cert7.db` and `key3.db` files from your user profile directory to a backup directory. This is a safety measure: these files shouldn't be needed again. Once the following steps are successfully completed, and you have used `signtool -l` to verify that the upgraded `cert7.db` file has the right certificates, you can discard these backup copies.
7. Copy Navigator's `cert5.db` and `key.db` files to your Communicator user-profile directory.
8. Restart Communicator. It automatically upgrades the older database files, including their contents.
9. Click the Security Info button on the Communicator toolbar, then click Yours under Certificates in the left panel.
10. Click Import a Certificate and give the database password.
11. Select a certificate file to open and give the certificate's password.

12. Repeat steps 9 and 10 for each certificate to be re-imported.

**The certificate needed to sign an object is in the certificate database, but Netscape Signing Tool's -l and -k options report "Unable to find issuer certificate" or "Unknown user" errors.**

Netscape Signing Tool 1.x reads the `cert7.db` files used by Communicator 4.x. Normally, `cert7.db` files record a certificate's complete certificate chain information using the PKCS #7 cryptographic message-syntax standard. However, Navigator 3.x wasn't designed to properly use this standard (it wasn't in wide use yet). Therefore, certificates used by Communicator 4.x that were originally imported with Navigator 3.x may not have all the certificate chain information required for object signing.

In the case of VeriSign Class 2 or Class 3 certificates, the missing portion of the chain is the intermediate node, since the root is provided with Communicator by default and the leaf is included in the existing certificate information. To get the intermediate portion, use Communicator 4.x and click these links for VeriSign Class 2 or VeriSign Class 3 certificate authority updates.

**When trying to read a JAR file into Communicator 4.05 the error message "Inconsistent files in manifest" appears in the Java console.**

Communicator 4.05 (and only that version) has a bug that makes it sensitive to the case of the JAR manifest's filename as stored in the META-INF directory. Version 4.05 requires the filename to be all lowercase. Although the JAR specification calls for case insensitivity and Netscape Signing Tool does generate a lowercase filename, an uppercase filename can appear if another tool is used to create the JAR, or case translation occurs on the Windows platform.

This problem can be repaired by re-signing the JAR with Netscape Signing Tool, or by unzipping the file and changing `MANIFEST.MF` to `manifest.mf`.

**How can users change the nicknames of their own certificates?**

For convenience, users may want to shorten the nicknames of some of the certificates they use. While certificates cannot be renamed directly, it may be possible to replace them and give the replacement a new name.

Note that this process can present a risk because it requires the user to delete a certificate before replacing it, and if the replacement fails and there is no backup certificate, the certificate is lost.

1. Click the Security Info button on Communicator's toolbar.
2. Click Yours under Certificates and select a certificate to rename.
3. Click Export and save a PKCS #12 copy of the certificate to a safe location (if no copy already exists). This copy is needed if replacement fails.
4. Click Delete and remove the certificate from the certificate database. Note that the certificate's corresponding private key isn't deleted, just the certificate itself.
5. Retrieve the certificate again from the Certificate Authority. The specific procedure for doing this depends on the Certificate Authority being used. Be very sure that the replacement is the correct one.
6. Enter a new name for the certificate when downloading it.

**Note:** The Export and Import buttons in the Security Info dialog box can't be used to change certificate nicknames. These functions can affect only a certificate's exported PKCS #12 filename.

**When I click the Security icon in the Communicator toolbar, click Yours under Certificates, select my object-signing certificate, and click Verify, Communicator informs me that the certificate is not valid. Why?**

This is a common occurrence. The Verify button works with S/MIME certificates only. It does not work with object-signing certificates.

To verify an object-signing certificate, use

```
signtool -l -k nickname
```

where *nickname* is the nickname of the certificate you want to verify.

**I get the following error when trying to create a test certificate:**

```
failure authenticating to key database .: Security I/O error.
```

This error typically means that you have not yet set a password for the certificate database. To set the password for the Communicator database, click the Security icon in the Communicator toolbar, click Passwords, and click Set Password to create a password.

**Objects to be signed will be stored and used long-term, well after the certificates used for signing have expired. Will signed objects still be trusted even after their object-signing certificates have expired?**

Although certificates expire, valid signatures do not. Signature validation is based on the date of the signature rather than the time verification occurs. If a certificate chain was valid at signing, Communicator will continue to recognize that signature even after certificates in that chain expire. Note that this would not be true, however, if an object was signed using the `-z` option which omits the original timestamp and forces validation to rely on the current status of the certificate chain.





# SSL Strength Tool

SSL Strength Tool is a command-line tool that connects to an SSL server and reports back the encryption cipher and strength used for the connection.

This appendix has the following sections:

- Availability (page 746)
- Syntax (page 746)
- Usage (page 747)
- Examples (page 748)

## Availability

SSL Strength Tool is available on the following platforms:

- HP-UX 11.0
- Solaris 2.5.1
- Windows NT 4.0
- AIX 4.1

# Syntax

```
sslstrength hostname[:port]
    [ciphers=ciphercode(s)]
    [verbose]
    [policy=export|domestic]
```

This form of the command returns a list of enabled ciphers on the client, then attempts to connect to the named SSL host, on the specified port. If the connection is successful, it returns information about the negotiated encryption strength.

```
sslstrength ciphers
```

This form of the command returns a list of the possible ciphers. A letter in the first column of the output is the code used by the `ciphers=` option. Pass any number of cipher codes to the `ciphers=` argument to identify the cipher preferences.

## Options and Arguments

The SSL Strength Tool command options and their arguments are defined as follows:  
Table 25.3

Options and Arguments	
hostname	Required. Identifies the SSL server to which to connect.
port	Optional. Identifies a specific port on the specified SSL server to which to connect. If not specified, defaults to the standard HTTPS port, 443.
ciphers=	Optional. Turns on the cipher preferences corresponding to the specified cipher codes, and turns off all other cipher preferences.
To obtain the list of cipher character codes, execute the special form of the command: sslstrength ciphers.	

Table 25.3

<code>verbose</code>	Optional. Turns on the verbose form of command output, which provides additional information about the progress of the connection.
<code>policy=</code>	Optional. Sets your policy regarding which ciphers can be permitted. Restricts the available ciphers to the same set used by Netscape Communicator for domestic or export versions (to comply with federal export restrictions).  The value can be <code>export</code> or <code>domestic</code> . If not specified, defaults to <code>domestic</code> .

# Usage

During an SSL handshake, the client sends the server a list of the ciphers it can use. The server chooses one of the ciphers based on its cipher policies, and notifies the client of which one to use.

When you issue the `sslstrength` command, the tool first prints the list of ciphers enabled on the client. It then connects to an SSL server and reports back the following information:

- The bulk encryption algorithm selected
- The key size selected
- The secret key size
- Information about the SSL server certificate, including:
  - The issuer subject name
  - The certificate subject name
  - The validity period

## Restricting Ciphers

You can selectively enable or disable specific ciphers on the client, to determine what strength of connection is used for those ciphers. Use the `policy=` or `ciphers=` option to restrict which ciphers are available.

- To restrict the available ciphers to the same set used by Communicator for exportable or domestic versions, set the `policy=` option to either `domestic` or `export`. In an exportable client, only those ciphers that are valid for export are enabled.
- To further restrict the ciphers available, use the `ciphers=` option. The argument to this option is a string of characters, where each single character represents a cipher. For example, `ciphers=bfi` turns on the cipher preferences corresponding to the codes `b`, `f`, and `i`. It turns off all other cipher preferences.

To obtain the list of cipher character codes, execute this command:

```
sslstrength ciphers
```

## Export Policy and Step-up

Some institutions, such as banks, may be qualified to obtain a special “step-up” certificate (also known as a “global server ID”) that allows the server to override export policy. When this certificate is installed in the server, it allows an export client that has step-up capability to renegotiate the SSL cipher and use domestic-strength encryption.

A connection that steps up starts out with 40-bit encryption, then, upon encountering a `change-cipher-spec` handshake, changes to 128-bit encryption. To check whether a client has stepped up correctly upon encountering a step-up certificate, check that it is using export policy, and that the secret key size is 128 bits.

## Examples

The following examples show the output from various `sslstrength` commands.

## Example 1

This example shows output from a command that allows all options to default.

```
sslstrength myhost.netscape.com
```

```
Using domestic policy
```

```
Connecting to myhost.netscape.com:443
```

```
Using all ciphersuites usually found in client
```

```
Your Cipher preference:
```

id	CipherName		Domestic	Export
a	SSL_EN_RC4_128_WITH_MD5	(ssl2)	Yes	No
b	SSL_EN_RC2_128_CBC_WITH_MD5	(ssl2)	Yes	No
c	SSL_EN_DES_192_EDE3_CBC_WITH_MD5	(ssl2)	Yes	No
d	SSL_EN_DES_64_CBC_WITH_MD5	(ssl2)	Yes	No
e	SSL_EN_RC4_128_EXPORT40_WITH_MD5	(ssl2)	Yes	Yes
f	SSL_EN_RC2_128_CBC_EXPORT40_WITH_MD5	(ssl2)	Yes	Yes
i	SSL_RSA_WITH_RC4_128_MD5	(ssl3)	Yes Step-up	only
j	SSL_RSA_WITH_3DES_EDE_CBC_SHA	(ssl3)	Yes Step-up	only
k	SSL_RSA_WITH_DES_CBC_SHA	(ssl3)	Yes	No
l	SSL_RSA_EXPORT_WITH_RC4_40_MD5	(ssl3)	Yes	Yes
m	SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	(ssl3)	Yes	Yes
o	SSL_RSA_WITH_NULL_MD5	(ssl3)	Yes	Yes
p	SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA	(ssl3)	Yes	No
q	SSL_RSA_FIPS_WITH_DES_CBC_SHA	(ssl3)	Yes	No

```
SSL Connection Status
```

```
Cipher:          RC4
```

```
Key Size:        128
```

```
Secret Key Size: 128
```

```
Issuer:          OU=Secure Server Certification Authority, O="RSA Data  
Security, Inc.", C=US
```

```
Subject:         CN=myhost.netscape.com, OU=E-Store Merchant Server,  
O=Netscape Communications Corp., L=Mountain View, ST=California, C=US
```

```
Valid:           from Fri Oct 02, 1998 to Sat Oct 02, 1999
```

## Example 2

This example shows output from a command that limits the client to three ciphers.

```
sslstrength myhost.netscape.com ciphers=jkl
```

```
Using domestic policy
```

```
Connecting to myhost.netscape.com:443
```

```
Your Cipher preference:
```

```
id      CipherName                                Domestic      Export
j      SSL_RSA_WITH_3DES_EDE_CBC_SHA              (ssl3)        Yes Step-up only
k      SSL_RSA_WITH_DES_CBC_SHA                   (ssl3)        Yes      No
l      SSL_RSA_EXPORT_WITH_RC4_40_MD5             (ssl3)        Yes      Yes
SSL Connection Status
Cipher:          3DES-EDE-CBC
Key Size:        168
Secret Key Size: 168
Issuer:          OU=Secure Server Certification Authority, O="RSA Data
Security, Inc.", C=US
Subject:         CN=myhost.netscape.com, OU=E-Store Merchant Server,
O=Netscape Communications Corp., L=Mountain View, ST=California, C=US
Valid:          from Fri Oct 02, 1998 to Sat Oct 02, 1999
```

### Example 3

This example shows output from a command that sets the client's policy to enable standard export ciphers.

```
sslstrength myhost.netscape.com policy=export
```

```
Using export policy
Connecting to myhost.netscape.com:443
Using all ciphersuites usually found in client
Your Cipher preference:
id      CipherName                                Domestic      Export
e      SSL_EN_RC4_128_EXPORT40_WITH_MD5           (ssl2)        Yes      Yes
f      SSL_EN_RC2_128_CBC_EXPORT40_WITH_MD5       (ssl2)        Yes      Yes
i      SSL_RSA_WITH_RC4_128_MD5                   (ssl3)        Yes Step-up only
j      SSL_RSA_WITH_3DES_EDE_CBC_SHA              (ssl3)        Yes Step-up only
l      SSL_RSA_EXPORT_WITH_RC4_40_MD5             (ssl3)        Yes      Yes
m      SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5         (ssl3)        Yes      Yes
o      SSL_RSA_WITH_NULL_MD5                      (ssl3)        Yes      Yes
SSL Connection Status
Cipher:          RC4-40
Key Size:        128
Secret Key Size: 40
Issuer:          OU=Secure Server Certification Authority, O="RSA Data
Security, Inc.", C=US
Subject:         CN=myhost.netscape.com, OU=E-Store Merchant Server,
O=Netscape Communications Corp., L=Mountain View, ST=California, C=US
Valid:          from Fri Oct 02, 1998 to Sat Oct 02, 1999
```

# SSL Debugging Tool

SSL Debugging Tool is an SSL-aware command-line proxy. It watches TCP connections and displays the data going by. If a connection is SSL, the data display includes interpreted SSL records and handshaking information.

This appendix has the following sections:

- Availability (page 751)
- Description (page 752)
- Syntax (page 752)
- Examples (page 754)
- Usage Tips (page 764)

## Availability

This tool is available for Solaris 2.5.1 (SunOS 5.5.1) and Windows NT 4.0.

## Description

The `ssltap` command opens a socket on a rendezvous port and waits for an incoming connection from the client side. Once this connection arrives, the tool makes another connection to the specified host name and port on the server side. It passes any data sent by the client to the server and vice versa. The tool also displays the data to the shell window from which it was called. It can do this for plain HTTP connections or any TCP protocol, as well as for SSL streams, as described here.

The tool cannot and does not decrypt any encrypted message data. You use the tool to look at the plain text and binary data that are part of the handshake procedure, before the secure connection is established.

## Syntax

To run SSL Debugging Tool, type this command in a command shell:

```
ssltap [-vhfsxl] [-p port] hostname:port
```

## Options

The command does not require any options other than *hostname:port*, but you normally use them to control the connection interception and output. The options for the command are the following:

---

-v	Print a version string for the tool.
-h	Turn on hex/ASCII printing. Instead of outputting raw data, the command interprets each record as a numbered line of hex values, followed by the same data as ASCII characters. The two parts are separated by a vertical bar. Nonprinting characters are replaced by dots.

---



---

<code>-f</code>	<p>Turn on fancy printing. Output is printed in colored HTML. Data sent from the client to the server is in blue; the server's reply is in red. When used with looping mode, the different connections are separated with horizontal lines. You can use this option to upload the output into a browser.</p>																						
<code>-s</code>	<p>Turn on SSL parsing and decoding. The tool does not automatically detect SSL sessions. If you are intercepting an SSL connection, use this option so that the tool can detect and decode SSL structures.</p> <p>If the tool detects a certificate chain, it saves the DER-encoded certificates into files in the current directory. The files are named <code>cert.0x</code>, where <code>x</code> is the sequence number of the certificate.</p> <p>If the <code>-s</code> option is used with <code>-h</code>, two separate parts are printed for each record: the plain hex/ASCII output, and the parsed SSL output.</p>																						
<code>-x</code>	<p>Turn on hex/ASCII printing of undecoded data inside parsed SSL records. Used only with the <code>-s</code> option. This option uses the same output format as the <code>-h</code> option.</p>																						
<code>-l</code>	<p>Turn on looping; that is, continue to accept connections rather than stopping after the first connection is complete.</p>																						
<code>-p port</code>	<p>Change the default rendezvous port (1924) to another port. The following are well-known port numbers:</p> <table> <tr><td>HTTP</td><td>80</td></tr> <tr><td>HTTPS</td><td>443</td></tr> <tr><td colspan="2"> </td></tr> <tr><td>SMTP</td><td>25</td></tr> <tr><td>FTP</td><td>21</td></tr> <tr><td colspan="2"> </td></tr> <tr><td>IMAP</td><td>143</td></tr> <tr><td>IMAPS</td><td>993 (IMAP over SSL)</td></tr> <tr><td colspan="2"> </td></tr> <tr><td>NNTP</td><td>119</td></tr> <tr><td>NNTPS</td><td>563 (NNTP over SSL)</td></tr> </table>	HTTP	80	HTTPS	443			SMTP	25	FTP	21			IMAP	143	IMAPS	993 (IMAP over SSL)			NNTP	119	NNTPS	563 (NNTP over SSL)
HTTP	80																						
HTTPS	443																						
SMTP	25																						
FTP	21																						
IMAP	143																						
IMAPS	993 (IMAP over SSL)																						
NNTP	119																						
NNTPS	563 (NNTP over SSL)																						

---

## Examples

You can use SSL Debugging Tool to intercept any connection information. Although you can run the tool at its most basic by issuing the `ssltap` command with no options other than *hostname:port*, the information you get in this way is not very useful.

For example, assume your development machine is called `intercept`. The simplest way to use the debugging tool is to execute the following command from a command shell:

```
ssltap www.netscape.com:80
```

The program waits for an incoming connection on the default port 1924. In your browser window, enter the URL `http://intercept:1924`. The browser retrieves the requested page from the server at `www.netscape.com`, but the page is intercepted and passed on to the browser by the debugging tool on `intercept`.

On its way to the browser, the data is printed to the command shell from which you issued the command. Data sent from the client to the server is surrounded by the following symbols:

```
--> [ data ]
```

Data sent from the server to the client is surrounded by the following symbols:

```
<-- [ data ]
```

The raw data stream is sent to standard output and is not interpreted in any way. This can result in peculiar effects, such as sounds, flashes, and even crashes of the command shell window. To output a basic, printable interpretation of the data, use the `-h` option, or, if you are looking at an SSL connection, the `-s` option.

You will notice that the page you retrieved looks incomplete in the browser. This is because, by default, the tool closes down after the first connection is complete, so the browser is not able to load images. To make the tool continue to accept connections, switch on looping mode with the `-l` option.

The following examples show the output from commonly used combinations of options.

## Example I

The `s` and `x` options in this example turn on SSL parsing and show undecoded values in hex/ASCII format. The output is routed to a text file.

### Command

```
ssltap.exe -sx -p 444 interzone.mcom.com:443 > sx.txt
```

### Output

```
Connected to interzone.mcom.com:443
--> [
alloclen = 66 bytes
  [ssl2] ClientHelloV2 {
    version = {0x03, 0x00}
    cipher-specs-length = 39 (0x27)
    sid-length = 0 (0x00)
    challenge-length = 16 (0x10)
    cipher-suites = {
      (0x010080) SSL2/RSA/RC4-128/MD5
      (0x020080) SSL2/RSA/RC4-40/MD5
      (0x030080) SSL2/RSA/RC2CBC128/MD5
      (0x040080) SSL2/RSA/RC2CBC40/MD5
      (0x060040) SSL2/RSA/DES64CBC/MD5
      (0x0700c0) SSL2/RSA/3DES192EDE-CBC/MD5
      (0x000004) SSL3/RSA/RC4-128/MD5
      (0x00ffe0) SSL3/RSA-FIPS/3DES192EDE-CBC/SHA
      (0x00000a) SSL3/RSA/3DES192EDE-CBC/SHA
      (0x00ffe1) SSL3/RSA-FIPS/DES64CBC/SHA
      (0x000009) SSL3/RSA/DES64CBC/SHA
      (0x000003) SSL3/RSA/RC4-40/MD5
      (0x000006) SSL3/RSA/RC2CBC40/MD5
    }
    session-id = { }
    challenge = { 0xec5d 0x8edb 0x37c9 0xb5c9 0x7b70 0x8fe9 0xd1d3
0x2592 }
  }
]
<-- [
SSLRecord {
  0: 16 03 00 03 e5 |.....
  type = 22 (handshake)
  version = { 3,0 }
  length = 997 (0x3e5)
  handshake {
    0: 02 00 00 46 |...F
```

```

    type = 2 (server_hello)
    length = 70 (0x000046)
    ServerHello {
        server_version = {3, 0}
        random = {...}
0: 77 8c 6e 26 6c 0c ec c0 d9 58 4f 47 d3 2d 01 45 |
wEn&1.i..XOG.-.E
    10: 5c 17 75 43 a7 4c 88 c7 88 64 3c 50 41 48 4f 7f |
\..uCSL.Ç.d&lt;PAHO.
        session ID = {
            length = 32
            contents = {...}
0: 14 11 07 a8 2a 31 91 29 11 94 40 37 57 10 a7 32 |
..."*1.)..@7W.$2
    10: 56 6f 52 62 fe 3d b3 65 b1 e4 13 0f 52 a3 c8 f6 |
VoRbp=³e±...RfÈ.
    }
    cipher_suite = (0x0003) SSL3/RSA/RC4-40/MD5
    }
0: 0b 00 02 c5 |...Å
    type = 11 (certificate)
    length = 709 (0x0002c5)
    CertificateChain {
        chainlength = 706 (0x02c2)
        Certificate {
            size = 703 (0x02bf)
            data = { saved in file 'cert.001' }
        }
    }
0: 0c 00 00 ca |....
    type = 12 (server_key_exchange)
    length = 202 (0x0000ca)
0: 0e 00 00 00 |....
    type = 14 (server_hello_done)
    length = 0 (0x000000)
    }
}
]
--> [
SSLRecord {
0: 16 03 00 00 44 |....D
    type = 22 (handshake)
    version = { 3,0 }
    length = 68 (0x44)
    handshake {
0: 10 00 00 40 |...@
    type = 16 (client_key_exchange)
    length = 64 (0x000040)
    ClientKeyExchange {
        message = {...}

```

```

    }
  }
}
]
--> [
SSLRecord {
  0: 14 03 00 00 01 |.....
  type    = 20 (change_cipher_spec)
  version = { 3,0 }
  length  = 1 (0x1)
  0: 01 |.
}
SSLRecord {
  0: 16 03 00 00 38 |....8
  type    = 22 (handshake)
  version = { 3,0 }
  length  = 56 (0x38)
          < encrypted >
}
]
<-- [
SSLRecord {
  0: 14 03 00 00 01 |.....
  type    = 20 (change_cipher_spec)
  version = { 3,0 }
  length  = 1 (0x1)
  0: 01 |.
}
]
<-- [
SSLRecord {
  0: 16 03 00 00 38 |....8
  type    = 22 (handshake)
  version = { 3,0 }
  length  = 56 (0x38)
          < encrypted >
}
]
--> [
SSLRecord {
  0: 17 03 00 01 1f |.....
  type    = 23 (application_data)
  version = { 3,0 }
  length  = 287 (0x11f)
          < encrypted >
}
]
<-- [
SSLRecord {
  0: 17 03 00 00 a0 |....

```

```

        type      = 23 (application_data)
        version = { 3,0 }
        length   = 160 (0xa0)
                < encrypted >
    }
  ]
  <-- [
  SSLRecord {
0: 17 03 00 00  df                                |....f
    type      = 23 (application_data)
    version = { 3,0 }
    length   = 223 (0xdf)
            < encrypted >
  }
  SSLRecord {
    0: 15 03 00 00 12                                |.....
    type      = 21 (alert)
    version = { 3,0 }
    length   = 18 (0x12)
            < encrypted >
  }
  ]
  Server socket closed.

```

## Example 2

The `-s` option turns on SSL parsing. Because the `-x` option is not used in this example, undecoded values are output as raw data. The output is routed to a text file.

## Command

```
ssltap.exe -s -p 444 interzone.mcom.com:443 > s.txt
```

## Output

```

Connected to interzone.mcom.com:443
--> [
alloclen = 63 bytes
  [ssl2] ClientHelloV2 {
    version = {0x03, 0x00}
    cipher-specs-length = 36 (0x24)
    sid-length = 0 (0x00)
    challenge-length = 16 (0x10)
    cipher-suites = {

```

```

        (0x010080) SSL2/RSA/RC4-128/MD5
        (0x020080) SSL2/RSA/RC4-40/MD5
        (0x030080) SSL2/RSA/RC2CBC128/MD5
        (0x060040) SSL2/RSA/DES64CBC/MD5
        (0x0700c0) SSL2/RSA/3DES192EDE-CBC/MD5
        (0x000004) SSL3/RSA/RC4-128/MD5
        (0x00ffe0) SSL3/RSA-FIPS/3DES192EDE-CBC/SHA
        (0x00000a) SSL3/RSA/3DES192EDE-CBC/SHA
        (0x00ffe1) SSL3/RSA-FIPS/DES64CBC/SHA
        (0x000009) SSL3/RSA/DES64CBC/SHA
        (0x000003) SSL3/RSA/RC4-40/MD5
    }
    session-id = { }
    challenge = { 0x713c 0x9338 0x30e1 0xf8d6 0xb934 0x7351 0x200c
0x3fd0 }
}
<-- [
SSLRecord {
    type      = 22 (handshake)
    version = { 3,0 }
    length    = 997 (0x3e5)
    handshake {
        type = 2 (server_hello)
        length = 70 (0x000046)
        ServerHello {
            server_version = {3, 0}
            random = {...}
            session ID = {
                length = 32
                contents = {...}
            }
            cipher_suite = (0x0003) SSL3/RSA/RC4-40/MD5
        }
    }
    type = 11 (certificate)
    length = 709 (0x0002c5)
    CertificateChain {
        chainlength = 706 (0x02c2)
        Certificate {
            size = 703 (0x02bf)
            data = { saved in file 'cert.001' }
        }
    }
    type = 12 (server_key_exchange)
    length = 202 (0x0000ca)
    type = 14 (server_hello_done)
    length = 0 (0x000000)
}
}
]
--> [

```

```

SSLRecord {
    type      = 22 (handshake)
    version   = { 3,0 }
    length    = 68 (0x44)
    handshake {
        type   = 16 (client_key_exchange)
        length = 64 (0x000040)
        ClientKeyExchange {
            message = {...}
        }
    }
}
]
--> [
SSLRecord {
    type      = 20 (change_cipher_spec)
    version   = { 3,0 }
    length    = 1 (0x1)
}
SSLRecord {
    type      = 22 (handshake)
    version   = { 3,0 }
    length    = 56 (0x38)
    < encrypted >
}
]
<-- [
SSLRecord {
    type      = 20 (change_cipher_spec)
    version   = { 3,0 }
    length    = 1 (0x1)
}
]
<-- [
SSLRecord {
    type      = 22 (handshake)
    version   = { 3,0 }
    length    = 56 (0x38)
    < encrypted >
}
]
--> [
SSLRecord {
    type      = 23 (application_data)
    version   = { 3,0 }
    length    = 287 (0x11f)
    < encrypted >
}
]
[

```



```

SSLRecord {
    type      = 23 (application_data)
    version = { 3,0 }
    length    = 160 (0xa0)
              < encrypted >
}
]
<-- [
SSLRecord {
    type      = 23 (application_data)
    version = { 3,0 }
    length    = 223 (0xdf)
              < encrypted >
}
SSLRecord {
    type      = 21 (alert)
    version = { 3,0 }
    length    = 18 (0x12)
              < encrypted >
}
]
Server socket closed.

```

## Example 3

In this example, the `-h` option turns hex/ASCII format. There is no SSL parsing or decoding. The output is routed to a text file.

## Command

```
ssltap.exe -h -p 444 interzone.mcom.com:443 > h.txt
```

## Output

```

Connected to interzone.mcom.com:443
--> [
    0: 80 40 01 03  00 00 27 00  00 00 10 01  00 80 02 00  |
..@.....
    10: 80 03 00 80  04 00 80 06  00 40 07 00  c0 00 00 04  |
.....@.....
    20: 00 ff e0 00  00 0a 00 ff  e1 00 00 09  00 00 03 00  |
.....á.....
    30: 00 06 9b fe  5b 56 96 49  1f 9f ca dd  d5 ba b9 52  |
...>þ[V.I.Ÿ...°¹R
    40: 6f 2d                                     |o-
]

```

```

<-- [
  0: 16 03 00 03 e5 02 00 00 46 03 00 7f e5 0d 1b 1d |
  .....F.....
  10: 68 7f 3a 79 60 d5 17 3c 1d 9c 96 b3 88 d2 69 3b |
h.:y'...&lt;:æ.³.Öi;
  20: 78 e2 4b 8b a6 52 12 4b 46 e8 c2 20 14 11 89 05 | x.K.|R.KFè.
  ..%.
  30: 4d 52 91 fd 93 e0 51 48 91 90 08 96 c1 b6 76 77 |
MR.Ý..QH.....¶vw
  40: 2a f4 00 08 a1 06 61 a2 64 1f 2e 9b 00 03 00 0b |
*ô...j.açd...>....
  50: 00 02 c5 00 02 c2 00 02 bf 30 82 02 bb 30 82 02 |
  ..Å.....0...0..
  60: 24 a0 03 02 01 02 02 02 01 36 30 0d 06 09 2a 86 | $
  .....60...*.
  70: 48 86 f7 0d 01 01 04 05 00 30 77 31 0b 30 09 06 |
H.÷.....0wl.0..
  80: 03 55 04 06 13 02 55 53 31 2c 30 2a 06 03 55 04 |
.U...US1,0*..U.
  90: 0a 13 23 4e 65 74 73 63 61 70 65 20 43 6f 6d 6d | ..#Netscape
Comm
  a0: 75 6e 69 63 61 74 69 6f 6e 73 20 43 6f 72 70 6f | unications
Corpo
  b0: 72 61 74 69 6f 6e 31 11 30 0f 06 03 55 04 0b 13 |
rationl.0...U...
  c0: 08 48 61 72 64 63 6f 72 65 31 27 30 25 06 03 55 |
.Hardcorel'0%..U
  d0: 04 03 13 1e 48 61 72 64 63 6f 72 65 20 43 65 72 | ....Hardcore
Cer
  e0: 74 69 66 69 63 61 74 65 20 53 65 72 76 65 72 20 | tificate
Server
  f0: 49 49 30 1e 17 0d 39 38 30 35 31 36 30 31 30 33 |
II0...9805160103
<additional data lines>
]
<additional records in same format>
Server socket closed.

```

## Example 4

In this example, the `-s` option turns on SSL parsing, and the `-h` options turns on hex/ASCII format. Both formats are shown for each record. The output is routed to a text file.

## Command

```
ssltap.exe -hs -p 444 interzone.mcom.com:443 > hs.txt
```

## Output

```
Connected to interzone.mcom.com:443
--> [
    0: 80 3d 01 03  00 00 24 00  00 00 10 01  00 80 02 00  |
.=....$.
    10: 80 03 00 80  04 00 80 06  00 40 07 00  c0 00 00 04  |
.....@
    20: 00 ff e0 00  00 0a 00 ff  e1 00 00 09  00 00 03 03  |
.....á
    30: 55 e6 e4 99  79 c7 d7 2c  86 78 96 5d  b5 cf e9
|U..™yÇ×,.x.]üİé
alloclen = 63 bytes
[ssl2] ClientHelloV2 {
    version = {0x03, 0x00}
    cipher-specs-length = 36 (0x24)
    sid-length = 0 (0x00)
    challenge-length = 16 (0x10)
    cipher-suites = {
        (0x010080) SSL2/RSA/RC4-128/MD5
        (0x020080) SSL2/RSA/RC4-40/MD5
        (0x030080) SSL2/RSA/RC2CBC128/MD5
        (0x040080) SSL2/RSA/RC2CBC40/MD5
        (0x060040) SSL2/RSA/DES64CBC/MD5
        (0x0700c0) SSL2/RSA/3DES192EDE-CBC/MD5
        (0x000004) SSL3/RSA/RC4-128/MD5
        (0x00ffe0) SSL3/RSA-FIPS/3DES192EDE-CBC/SHA
        (0x00000a) SSL3/RSA/3DES192EDE-CBC/SHA
        (0x00ffe1) SSL3/RSA-FIPS/DES64CBC/SHA
        (0x000009) SSL3/RSA/DES64CBC/SHA
        (0x000003) SSL3/RSA/RC4-40/MD5
    }
    session-id = { }
    challenge = { 0x0355 0xe6e4 0x9979 0xc7d7 0x2c86 0x7896 0x5db
0xcfe9 }
}
]
<additional records in same formats>
Server socket closed.
```

## Usage Tips

- When SSL restarts a previous session, it makes use of cached information to do a partial handshake. If you wish to capture a full SSL handshake, restart the browser to clear the session id cache.
- If you run the tool on a machine other than the SSL server to which you are trying to connect, the browser will complain that the host name you are trying to connect to is different from the certificate. If you are using the default BadCert callback, you can still connect through a dialog. If you are not using the default BadCert callback, the one you supply must allow for this possibility.

# Index

## A

- accelerators 198
- active logs
  - default file location 572
  - naming convention 573
  - See also* logging
- adding
  - new authentication instances 311, 317
    - relationship with enrollment forms 317
  - new jobs 383, 387
  - new policy rules 461, 468
- Administration Server 51
  - relationship to Netscape Console 51
  - relationship to server root 51
  - starting 52
    - from Netscape Console 52
    - from the command line 52
    - from the Windows NT Services panel 52
  - stopping 53
    - from Netscape Console 53
    - from the command line 53
    - from the Windows NT Services panel 53
- administrators
  - common tasks 58
  - deleting 181
  - designated group 146
  - modifying 175
    - group membership 179
    - login information 176
  - port used for operations 122
    - See also* ports
  - role defined 134
  - setting up 150
  - tools provided
    - CMS window 55
    - Netscape Console 48
- agents
  - authorizing remote key recovery 632
  - deleting 181
  - designated groups 147
  - forms for 563
  - locating forms and templates for 564
  - modifying 175
    - certificate information 178
    - group membership 179
    - login information 176
  - port used for operations 123
    - See also* ports
  - role defined 135
  - setting up 152
  - SSL client certificates for 137
  - See also* Agent Services interface
- Agent Services interface 541
  - for Certificate Manager agents 541
  - for Data Recovery Manager agents 543
  - for Registration Manager agents 542
  - how to access 544
  - URL for 123
  - who can access 541
- archive
  - See* backup
- archiving
  - rotated log files 576
  - users' encryption private keys 623
- ASCII to Binary tool 676
  - example 677
  - supported platforms 676
  - syntax 677
- Audit log
  - defined 569
  - how to configure 585
  - how to monitor 592
  - logging to Windows NT event log 594
  - See also* logging

- authentication
  - built-in modules 265
    - directory- and PIN-based 275
    - directory-based 268
    - manual 266
    - See also* PIN Generator tool
  - defined 259
  - developing custom plug-ins 333
    - API for 334
    - compiling 335
    - installing 335
    - samples 339
  - difference between plug-in implementation and instance 314
  - during certificate enrollment 265
  - during certificate renewal 282
  - during certificate revocation 283
  - for administrators 260
  - for agents 262
  - for end entities 265
  - managing from CMS window 310
  - subsystem architecture 330
    - how it works 331
- authentication instances
  - adding new 311, 317
    - relationship with enrollment forms 317
  - deleting 311, 322
  - how they're used 332
  - modifying 311, 322
  - naming 317
- authentication modules
  - deleting 313, 327
  - developing new 329
  - how they're used 332
  - registering new ones 312, 325
- Authority Key Identifier extension policy 432

## B

- backing up
  - CMS configuration 668
  - CMS data 668
- backup
  - defined 666

- guidelines for creating 666
  - reasons for creating 666
- base DN 661
- Basic Constraints extension policy 435
- Binary to ASCII tool 677
  - example 678
  - supported platforms 677
  - syntax 677
- buffered logging 574
- built-in plug-in modules
  - See* plug-in modules

## C

- CA signing certificate 184
  - changing trust settings of 250
  - deleting 249
  - getting a new one 232
  - nickname 185
  - renewing 239
  - viewing details of 247
- certificate chains
  - getting 559
  - installing in the certificate database 215
  - why you should install 255
- certificate database
  - how to manage 246
  - what it contains 246
  - where it's maintained 246
- Certificate Database tool 232, 240
- certificate enrollment
  - supported authentication mechanisms 536
  - supported request formats 536
- Certificate Enrollment Protocol (CEP) 613
- certificate issuance
  - to routers 613
    - an example 617
  - to servers 603
    - manual enrollment 604
    - Netscape 3.x servers 607
    - Netscape 4.x servers 612
- Certificate Manager

- configuring
  - for LDAP publishing 507
  - for publishing CA certificate 511
  - for publishing CRLs 527
  - for publishing end-entity certificates 513
  - publishing directory 508
  - SMTP settings for notifications 131, 132
  - to use separate SSL server certificates 224
  - to use specific ciphers 230
- connecting to a Data Recovery Manager 167
- enabling interaction with end entities 538
- enrollment forms for 556
- interface for agents 541
- key pairs and certificates
  - CA signing certificate 184
  - getting new ones 232
  - list of 184
  - renewing existing ones 239
  - SSL server certificate 186
- logging to Windows NT event log 594
- manual updates to publishing directory 520
- specifying IP address for 127
- what to do if not responding 116
- certificate renewal 619
  - authentication during 282
  - of client certificates 619
  - of server certificates 621
  - supported authentication mechanisms 536
  - supported request formats 536
- certificate request
  - result of policy processing 408
- certificate request formats 536
  - for enrollments 536
  - for key archival and recovery 537
  - for renewals 536
  - for revocations 537
- certificate revocation
  - authentication during 283
  - reasons for 525
  - supported authentication mechanisms 537
  - supported request formats 537
- certificates
  - expired
    - publishing to the directory 529
  - revocation reasons 525
  - revoking 523
- Certificate Setup Wizard 199
  - using to install certificate chains 215
  - using to install certificates 215
    - supported data formats 216
  - using to request certificates 200
- changing
  - CMS instance name 98, 99
    - character set for the name 96
    - format for the name 98
  - group members 179
  - port numbers 125
    - See also* ports
  - trust settings in certificates 250
    - why would you change 250
- changing passwords 107, 118
- checking CMS status 115
- ciphers
  - configuring 230
  - defined 228
  - list of 229
  - step-up program for international export
    - browsers 230
  - supported on the server side 228
  - which ones to choose 229
- classpath for adding plug-ins 335
- client certificate renewal 619
- CMS configuration
  - how to backup 668
  - See also* configuration file
- CMS data
  - how to backup
  - where it's stored 128
- CMS feature list 33
- CMS instance
  - changing the name 98, 99
    - character set for the name 96
    - format for the name 98
  - creating multiple instances 94
  - removing 99
  - viewing information 96

- file location 97
  - installation date 97
  - on/off status 98
  - security level 98
  - version number 98
- CMS watchdog 117
- CMS window
  - Configuration tab 58
  - configuring authentication 310
  - configuring jobs 382
  - configuring LDAP publishing 507, 515
  - configuring network settings 121
  - configuring policies 460
  - how to launch 63, 65
  - introduction 55
  - managing logs 578
  - Status tab 63
  - Tasks tab 57
  - using to manage policies 468
  - who can launch 65
- command-line utilities 673
  - ASCII to Binary 676
  - Binary to ASCII 677
  - dumpasn1 683
  - killproc tool 116, 675
  - location 676
  - PIN Generator 285
  - Pretty Print Certificate 678
  - Pretty Print CRL 681
  - some guidelines 675
  - summary table 673
- configuration
  - road map 83
  - ways to modify 71
- configuration file 67
  - copying from one instance to another 70
  - effects of installation on 68
  - format 73
  - format for localizable values 74
  - guidelines for editing 73
  - how subsystem-specific parameters are distinguished 73
  - location 71
  - name 67
  - sample 74
  - shared parameters 68
  - ways to modify
    - by editing the file 72
    - from CMS window 71
  - what is ignored by the server 73
  - what it controls 68
  - when created 67
- Configuration tab 58
  - tasks you can accomplish 58
- configuring logs 581
  - Audit log 585
  - Error log 583
  - System log 581
  - See also* logging 581
- connecting subsystems 141, 158
  - connection types 143
  - connectors 143
  - why would you do this 141
- constraints-specific policy modules 409
  - default revocation 411
  - DSA key 413
  - key algorithm 416
  - renewal validity 419
  - RSA key 422
  - validity 426
- conventions used in this book 25
- core features 33
- creating multiple CMS instances 94
- CRL Distribution Point extension 524
- CRL Distribution Point extension policy 438
- CRLs
  - defined 523
  - extensions 527
  - issuing or distribution points 524
  - publishing
    - configuring a Certificate Manager for 527
    - directory entry for publishing 527
    - directory for publishing 527
    - expired certificates 529
    - specifying allowed extensions 529
    - specifying interval 529
    - specifying signing algorithm 529



- things you need to know 526
- what happens to the old CRL 527
- supported versions 527
- updating manually 529
  - who's allowed to do this 530
- when automated updates take place 526
- who generates it 524

## D

- data formats for installing certificate chains 216
  - binary 216
  - text 217
- data formats for installing certificates 216
  - binary 216
  - text 217
- Data Recovery Manager
  - configuring
    - to use separate SSL server certificates 224
    - to use specific ciphers 230
  - connecting to a Certificate Manager 167
  - connecting to a Registration Manager 158
  - interface for agents 543
  - key pairs and certificates
    - getting new ones 232
    - list of 190
    - renewing existing ones 239
    - SSL server certificate 191
    - storage key pair 191
    - transport certificate 190
  - logging to Windows NT event log 594
  - setting up
    - key archival 641
    - key recovery 651
  - specifying IP address for 127
  - what to do if not responding 116
- default revocation policy 411
- deleting
  - authentication instances 311, 322
  - authentication modules 313, 327
  - certificates from the token 249
    - precaution 249
  - job modules 384, 398
  - jobs 383, 391

- policy modules 463, 481
- policy rules 461, 472
- privileged users 181
- rotated log files 575
- developing custom plug-ins
  - classpath 335
- developing plug-ins
  - for authentication 333
    - API 334
    - compiling 335
    - installing 335
    - samples 339
- directory
  - removing expired certificates from 358
  - schema for PINs 300
  - setting up for LDAP publishing 504
- directory-based authentication 268
  - user ID, password, and PIN 275
  - user ID and password 268
- distinguished name (DN)
  - base DN 661
  - components 660
  - defined 659
  - guidelines for choosing DNs 664
  - role in certificates 662
    - CA certificates 664
    - end-entity certificates 663
  - root DN 661
- documentation
  - conventions followed 25
  - where to find 27
- DSA key constraints policy 413
- dumpasn1 tool 683

## E

- end entities
  - enabling interaction with a Certificate Manager 538
  - enabling interaction with a Registration Manager 539
  - forms provided for 533
  - generating PINs for 299, 300

- locating forms and templates 555
- port used for operations 124
  - See also* ports
- publishing certificates to a directory 513, 518
- supported request formats 536
- end-entity certificates
  - renewal 619
  - revocation 621
- end-entity forms 553
  - for enrollment 555
  - for renewal 557
  - for retrieval 558
  - for revocation 557
- end-entity templates 561
- enrollment forms
  - for Certificate Managers 556
  - for end users 555
  - for object signing certificates 556
  - for Registration Managers 556
  - for servers 555
  - specifying authentication 317
- Error log
  - defined 568
  - how to configure 583
  - how to monitor 589
  - See also* logging
- event log
  - logging audit and system messages 594
- expired certificates
  - publishing them to the directory 529
  - removing from the directory 358
- extension-specific policy modules 431
  - authority key identifier 432
  - basic constraints 435
  - CRL distribution point 438
  - key usage 446
  - list of 431
  - Netscape certificate type 449
  - subject alternate name 453
  - subject key identifier 455
- external tokens
  - defined
  - installing 194

- viewing contents of 247

## F

- filenames
  - for active log files 573
  - for rotated log files 573
- flush interval for logs 574
- fonts used in this book 25
- forms
  - See* HTML forms

## G

- generating PINs for end entities 299, 300
- getting new certificates for subsystems 232
- groups
  - changing members 179
  - defined 146
  - for administrators 146
  - for agents 147
  - for trusted managers 149
  - where they're maintained 146
- guidelines
  - for creating backups 666
  - for restoring 667

## H

- hardware accelerators 198
- hardware tokens
  - See* external tokens
- host name
  - for mail server used for notifications 132
- how to check whether CMS is on or off 115
- how to search for keys 627
- HTML forms
  - for agents 541, 563, 564
  - for end entities 533, 555
  - for enrollment 555
  - for renewal 557
  - for retrieval 558

for revocation 557

## I

installation date 97

installing external hardware tokens 194

installing multiple CMS instances 94

internal database

- default host name 130

- precaution for changing the host name 130

- defined 128

- how to distinguish from other Directory

- Server instances 129

- name format 129

- schema 129

- what you shouldn't do 129

- what is it used for 128

- when installed 129

internal tokens

- viewing contents of 247

IP address 127

issuing certificates

- to routers 613

- an example 617

- to servers 603

- manual enrollment 604

- Netscape 3.x servers 607

- Netscape 4.x servers 612

## J

job modules

- deleting 384, 398

- registering new ones 384, 396

jobs

- adding new 383, 387

- built-in modules 346

- RenewalNotificationJob 346, 347

- RequestInQueueJob 346, 353

- UnpublishExpiredJob 347, 358

- compared to plug-in implementation 387

- configuration parameters 385

- created during installation 392

- deleting 383, 391

- managing from CMS window 382

- modifying 383, 391

- naming 387

- setting frequency 394

- specifying schedule for 363

- turning on scheduler 394

## K

key algorithm constraints policy 416

key archival 626

- how it works 627

- how keys are stored 627

- how to set up 641

- PKI setup required 624

- required format for requests 537

- where keys are stored 626

- why you should archive 626

key features 33

key recovery 629

- designated agents

- See key recovery agents

- how to set up 651

- interface for agents 631

- local vs. remote 632

key recovery agents

- passwords 630

- significance 630

- when specified the first time 630

- responsibilities 630

- role defined 630

Key Usage extension policy 446

killproc tool 116, 675

## L

LDAP publishing

- advantages 485

- configuring a Certificate Manager 507

- directory identity 508

- rules for CA certificate 511

- rules for end-entity certificates 513

- configuring a Registration Manager 515

- directory identity 515

- rules for end-entity certificates 518
- defined 486
- directory schema 499
  - for CRLs 501
  - for end-entity certificates 500
  - for the CA certificate 500
  - how to set up 504
- manual updates 520
  - when to do 520
  - who can do this 520
- See* CRLs
- linking subsystems
  - See* connecting subsystems
- list of
  - agent forms and templates
  - end-entity forms and templates
- local vs. remote key recovery 632
- location of
  - active log files
  - agent forms 564
  - CMS configuration file 71
  - CMS documentation 27
  - command-line utilities 676
  - end-entity forms 555
  - PIN Generator tool 286
  - rotated log files 575
- logging
  - buffered vs. unbuffered 574
  - configuring
    - Audit log 585
    - Error log 583
    - System log 581
  - log files
    - archiving rotated files 576
    - automatic deletion 575
    - automatic rotation 574
    - default location 572
    - location of rotated files 575
    - naming convention for active logs 573
    - naming convention for rotated logs 573
    - significance of deleting files 575
    - timing of rotation 575
- log levels 570
  - default selection 572

- how they're represented 571
  - how they relate to message categories 571
  - significance of choosing the right level 572
  - what it means 571
- managing from CMS window 578
- monitoring
  - Audit log 592
  - Error log 589
  - System log 587
  - using system tools in Windows NT 594
- parameters in the configuration file 580
- services that are logged 569
- types of logs 568
  - Audit 569
  - Error 568
  - System 568

## M

- mail server used for notifications 132
- managing
  - certificate database 246
  - job plug-in modules 396
  - policies 468
  - policy plug-in modules 479
  - privileged users 133
  - schedulable jobs 387
- manual authentication 266
- manually updating CRLs 529
  - who's allowed to do this 530
- message templates for notifications 370
- modifying
  - authentication instances 311, 322
  - jobs 383, 391
  - policy rules 461, 473
  - privileged user's group membership 179
  - privileged-user information 175
- m of n secret sharing 630
- monitoring logs 586
  - Audit log 592
  - Error log 589
  - System log 587
  - things you can monitor 586
  - using system tools in Windows NT 594

*See also* logging

## N

naming convention

- for active logs 573
- for CMS instances 96
- for internal database instances 129
- for rotated logs 573

Netscape Certificate Type extension policy 449

Netscape Console

- checking CMS status 115
- how to launch 53
  - in Unix 54
  - in Windows NT 54
- installing multiple CMS instances 94
- introduction 48
- opening CMS window 63
- relationship to Administration Server 51
- removing a CMS instances 99
- restarting Certificate Management System 113
- starting Administration Server 52
- starting Certificate Management System 107
- stopping Administration Server 53
- stopping Certificate Management System 111
- viewing CMS instance information 96

nickname

- for CA signing certificate 185
- for signing certificate 188
- for SSL server certificate 186, 188, 191
- for transport certificate 190

notifications

- configuring the mail server 131
  - host name 132
  - port 132
- customizing 370
  - templates 374
- event-driven 364
  - when certificates are issued 365
  - when new requests are queued 367
- sending renewal notifications to end entities 347
- to agents about pending requests 353
- to agents about unpublishing certificates 358

## O

object signing certificates

- how to enroll for 556

output templates

- for end-entity operations 561

## P

passwords

- changing cached 107, 118
- See also* single signon passwords

PIN Generator tool 285

- arguments 286
- delivering PINs to users 307
- directory schema requirements 300
  - changing 3.x directory schema 301
  - changing 4.x directory schema 303
- exit codes 298
- generating PINs 299
- how it works 292
- how PINs are stored in the directory 298
- output file 296
  - checking the directory-entry status 294
  - format 297
  - why should you use an output file 294
- overwriting existing PINs in the directory 290, 294
- syntax 286
- where to find 286

plug-in modules

- classpath for adding 335
- for authentication
  - developing new ones 333
  - UID, password, and PIN based 276
  - UID and password based 270
- for LDAP publishing
  - mapping certificates to directory entries 491
  - publishing certificates to directory entries 498
- for policy 409
  - authority key identifier extension 432
  - basic constraints extension 435
  - CRL distribution point extension 438

- default revocation 411
  - DSA key constraints 413
  - key algorithm constraints 416
  - key usage extension 446
  - managing 479
  - Netscape certificate type extension 449
  - renewal validity constraints 419
  - RSA key constraints 422
  - subject alternate name extension 453
  - subject key identifier extension 455
  - validity constraints 426
- for scheduling jobs
  - removal of expired certificates from
    - directory 358
  - renewal notifications to end entities 347
  - request-in-queue notifications to
    - agents 353
- policy
  - built-in plug-in modules 409
  - configuration parameters 463
  - constraints-specific modules 409
  - defined 402
  - extension-specific modules 431
  - managing 468
  - managing from CMS window 460
  - processor 408
    - how it applies rules 408
    - result of processing 408
    - when used 408
  - what can you use it for 402
- policy modules
  - deleting 463, 481
  - registering new ones 462, 479
- policy rules
  - adding new 461, 468
  - compared to policy implementation 466
  - configuration parameters 463
  - created during installation 474
  - defined 403
  - deleting 461, 472
  - how policy processor applies them 408
  - modifying 461, 473
  - naming 468
  - predicates in 404
  - reordering 462, 477
    - significance of ordering 477
  - See also* predicates
  - types of 403
  - what each rule does 403
- ports 121
  - changing numbers 125
  - for agent operations 123
  - for end-entity operations 124
    - turning on/off HTTP port 126
  - for remote administration 122
  - for the mail server used for notifications 132
  - how to choose numbers 122
    - in Unix 122
    - in Windows NT 122
- predicates
  - attributes for 406
  - expression support 404
    - operators for 404
  - sample expressions 406
  - what are they 404
  - why would you use 404
- Pretty Print Certificate tool 678
  - example 679
  - supported platforms 678
  - syntax 678
- Pretty Print CRL tool 681
  - example 681
  - supported platforms 681
  - syntax 681
- privileged users 133, 134
  - deleting 181
  - groups 146
  - modifying privileges 175
    - certificate information 178
    - group membership 179
    - login information 176
  - setting up 149
    - administrators 150
    - agents 152
    - trusted managers 158
  - types 134
    - administrators 134
    - agents 135
    - determining factor 134

- trusted manager 141
- types or roles 134
- CRLs
  - publishing
    - See also* LDAP publishing
- publishing
  - See* LDAP publishing

## R

- reasons for revoking certificates 525
- recovering users' private keys 629
- registering
  - authentication modules 312, 325
  - job modules 384, 396
  - policy modules 462, 479
- Registration Manager
  - configuring
    - for LDAP publishing 515
    - for publishing end-entity certificates 518
    - publishing directory 515
    - to use separate SSL server certificates 224
    - to use specific ciphers 230
  - connecting to another subsystem 158
  - enabling interaction with end entities 539
  - enrollment forms for 556
  - interface for agents 542
  - key pairs and certificates
    - getting new ones 232
    - list of 187
    - renewing existing ones 239
    - signing certificate 188
    - SSL server certificate 188
  - logging to Windows NT event log 594
  - specifying IP address for 127
  - what to do if not responding 116
- removing unwanted CMS instances 99
- renewal of certificates
  - See* certificate renewal
- renewal validity constraints policy 419
- renewing certificates of subsystems 239
- reordering policy rules 462, 477
  - significance of ordering 477
- request formats for certificates 536
- restarting
  - Certificate Management System 113
    - from Netscape Console 113
    - from the command line 115
- restore
  - defined 667
  - guidelines for 667
  - when to do 667
- revocation policy 411
- revoking certificates 523, 621
  - reasons 525
- road map to configuring subsystems 83
- roles
  - administrator 134
  - agent 135
  - determining factor 134
  - key recovery agents 630
  - trusted manager 141
- root DN 661
- rotated logs
  - naming convention 573
- rotating log files 574
  - archiving files 576
  - conserving disk space 575
  - how to set the time 575
- routers
  - getting certificates for 613, 617
  - port used for requesting 613
- RSA key constraints policy 422

## S

- samples
  - for authentication 339
- schedulable jobs
  - See* jobs
- secret sharing of storage key pair 630
- security level 98
- server's on/off status 115
- server certificate renewal 621

- server enrollment forms 555
- server instance
  - finding out details 96
- server name
  - changing 98
- server root
  - default for Unix 97
  - default for Windows NT 97
  - defined 97
  - how many on a single host 97
  - relationship with Administration Server 51
- setpin command 286
- setting up
  - key archival 641
  - key recovery 651
- setting up directory for LDAP publishing 504
  - See also* LDAP publishing
- signing certificate 188
  - changing trust settings of 250
  - deleting 249
  - getting a new one 232
  - nickname 188
  - renewing 239
  - viewing details of 247
- single signon password
  - changing cached passwords 107, 118
  - what it protects 106
  - when required 106
  - when specified 107
- SMTP settings 131, 132
- specifying IP address 127
- SSL server certificate 186, 188, 191
  - changing trust settings of 250
  - deleting 249
  - getting a new one 232
  - nickname 186, 188, 191
  - renewing 239
  - viewing details of 247
- starting
  - Administration Server 52
    - from Netscape Console 52
    - from the command line 52
  - from the Windows NT Service panel 52
  - Certificate Management System 106
    - from Netscape Console 107
    - from the command line 109
    - from the Windows NT Services panel 109
    - information required 106
  - Netscape Console 53
    - in Unix 54
    - in Windows NT 54
- Status tab 63
  - tasks you can accomplish 63
- stopping
  - Administration Server 53
    - from Netscape Console 53
    - from the command line 53
    - from the Windows NT Services panel 53
  - Certificate Management System 110
    - from Netscape Console 111
    - from the command line 112
    - from the Windows NT Services panel 112
- storage key pair 191
  - secret sharing 630
- stronger encryption for export browsers 230
- Subject Alternate Name extension policy 453
- Subject Key Identifier extension policy 455
- subordinate CA
  - enrollment forms for 556
- System log
  - defined 568
  - how to configure 581
  - how to monitor 587
  - logging to Windows NT event log 594
  - See also* logging

## T

- Tasks tab 57
  - tasks you can accomplish 57
- templates
  - for agents
    - location 564
  - for end entities
    - location 555



- for end-entity operations 561
- for notifications 370, 372
  - customizing 374
  - token list 374
- templates
  - for automated notifications 370
- timing log file deletion 576
- timing log rotation 575
- tokens
  - changing password of 198
  - deleting certificates from 249
  - external 193
    - See also* external tokens
  - internal 193
  - managing 197
  - viewing contents of 247
  - viewing which tokens are installed 197
  - what are they 192
- transport certificate 190
  - changing trust settings of 250
  - deleting 249
  - getting a new one 232
  - nickname 190
  - renewing 239
  - viewing details of 247
  - when used 629
- trusted managers
  - certificate for SSL client authentication 144
  - connectors for linking 143
  - deleting 181
  - designated group 149
    - access rights 149
  - modifying 175
    - certificate information 178
    - group membership 179
    - login information 176
  - role defined 141
  - setting up 158
- type styles used in this book 25

## U

- unbuffered logging 574
- uninstalling Certificate Management System 101

- from the command line 101
- using Windows NT Add/Remove Programs utility 102
- user enrollment forms 555
- user ID, password, and PIN based authentication 275
  - configurable parameters 276
  - module name 276
- user ID and password based authentication 268
  - configurable parameters 270
  - plug-in module name 270
- users
  - privileged 133

## V

- validity constraints policy 426
- version number 98
- viewing
  - contents of a token 247
- viewing CMS instance information 96

## W

- watchdog 117
- when the server was installed 97
- why should you revoke certificates 525
- Windows NT event log
  - logging audit and system messages 594
- wizard
  - See* Certificate Setup Wizard