

Plug-Ins Guide

SunTM ONE Certificate Server

Version 4.7

September 2002
816-5546-10
Second Edition

Copyright © 2002 Sun Microsystems, Inc. All rights reserved.

Sun, Sun Microsystems, and the Sun logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Federal Acquisitions: Commercial Software -- Government Users Subject to Standard License Terms and Conditions.

The product described in this document is distributed under licenses restricting its use, copying, distribution, and decompilation. No part of the product or this document may be reproduced in any form by any means without prior written authorization of the Sun Microsystems, Inc. and its licensors, if any.

THIS DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Some pre-existing portions:

Copyright © 1998,1999 by Jef Poskanzer <jef@acme.com>. All rights reserved. Copyright © 1996 by Jef Poskanzer <jef@acme.com>. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) "HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT "LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright © 2002 Sun Microsystems, Inc. Tous droits réservés.

Sun, Sun Microsystems, le Sun logo, et iPlanet sont des marques dposes ou des marques dposes registre de Sun Microsystems, Inc. aux Etats-Unis et d'autres pays.

Le produit dé crit dans ce document est distribué selon des conditions de licence qui en restreignent l'utilisation, la copie, la distribution et la décompilation.

Aucune partie de ce produit ni de ce document ne peut être reproduite sous quelque forme ou par quelque moyen que ce soit sans l'autorisation écrite préalable de Sun Microsystems, Inc., le cas échéant, de ses bailleurs de licence.

CETTE DOCUMENTATION EST FOURNIE "EN L'ÉTAT", ET TOUTES CONDITIONS EXPRESSES OU IMPLICITES, TOUTES REPRÉSENTATIONS ET TOUTES GARANTIES, Y COMPRIS TOUTE GARANTIE IMPLICITE D'APTITUDE À LA VENTE, OU À UN BUT PARTICULIER OU DE NON CONTREFAÇON SONT EXCLUES, EXCEPTÉ DANS LA MESURE OÙ DE TELLES EXCLUSIONS SERAIENT CONTRAIRES À LA LOI.

Contents

About This Guide	11
What's in This Guide	12
What You Should Already Know	13
Conventions Used in This Guide	14
Where to Go for Related Information	15
Chapter 1 Authentication Plug-in Modules	19
Overview of Authentication Modules	20
Manual Authentication	23
UidPwddirAuth Plug-in Module	24
Configuration Parameters of UidPwddirAuth	26
UidPwddirAuth Plug-in Module	30
Configuration Parameters of UidPwddirAuth	31
NISAuth Plug-in Module	37
Configuration Parameters of NISAuth	39
PortalEnroll Plug-in Module	44
Configuration Parameters of PortalAuth	47
SSOAuthentication Plug-In Module	52
Configuring SSOAuthentication	52
Certificate-Based Enrollment	53
Enrollment Forms	57
Customizing Enrollment Forms for Generating DSA Key Pairs	61
Generating Files Required By Third-Party Object Signing Tools	63
Chapter 2 Job Plug-in Modules	67
Overview of Job Plug-in Modules	67
RenewalNotificationJob Plug-in Module	69
Configuration Parameters of RenewalNotificationJob	70
RequestInQJob Plug-in Module	73
Configuration Parameters of RequestInQJob	74

UnpublishExpiredJob Plug-in Module	76
Configuration Parameters of UnpublishExpiredJob	78
Schedule for Executing Jobs	80
Customizing Notification Messages	81
Templates for Summary Notifications	81
Customizing Message Templates	83
Tokens Available in Message Templates	83
Tokens for Renewal Notification Messages	83
Tokens for Request In Queue Notification Messages	85
Tokens for Directory Update Notification Messages	85
 Chapter 3 Constraints Policy Plug-in Modules	87
Overview of Constraints-Specific Policy Modules	88
AttributePresentConstraints Plug-in Module	90
Configuration Parameters of AttributePresentConstraints	91
DSAKeyConstraints Plug-in Module	95
Configuration Parameters of DSAKeyConstraints	96
DSAKeyRule Rule	98
IssuerConstraints Plug-in Module	98
Configuration Parameters of IssuerConstraints	99
IssuerRule Rule	100
KeyAlgorithmConstraints Plug-in Module	101
Configuration Parameters of KeyAlgorithmConstraints	101
KeyAlgRule Rule	103
RenewalConstraints Plug-in Module	103
Configuration Parameters of RenewalConstraints	104
RenewalConstraintsRule Rule	105
RenewalValidityConstraints Plug-in Module	106
Configuration Parameters of RenewalValidityConstraints	107
DefaultRenewalValidityRule Rule	109
RevocationConstraints Plug-in Module	110
Configuration Parameters of RevocationConstraints	110
RevocationConstraintsRule Rule	111
RSAKeyConstraints Plug-in Module	112
Configuration Parameters of RSAKeyConstraints	112
RSAKeyRule Rule	114
SigningAlgorithmConstraints Plug-in Module	115
Configuration Parameters of SigningAlgorithmConstraints	116
SigningAlgRule Rule	118
SubCANameConstraints Plug-in Module	118
Configuration Parameters of SubCANameConstraints	119
SubCANameConstraints Rule	120
UniqueSubjectNameConstraints Plug-in Module	121

Configuration Parameters of UniqueSubjectNameConstraints	121
UniqueSubjectNameConstraints Rule	124
ValidityConstraints Plug-in Module	124
Configuration Parameters of ValidityConstraints	126
DefaultValidityRule Rule	128
 Chapter 4 Certificate Extension Plug-in Modules	131
Overview of Extension-Specific Policy Modules	132
AuthInfoAccessExt Plug-in Module	136
Configuration Parameters of AuthInfoAccessExt	138
AuthInfoAccessExt Rule	143
AuthorityKeyIdentifierExt Plug-in Module	144
Configuration Parameters of AuthorityKeyIdentifierExt	145
AuthorityKeyIdentifierExt Rule	147
BasicConstraintsExt Plug-in Module	147
Configuration Parameters of BasicConstraintsExt	148
BasicConstraintsExt Rule	150
CertificatePoliciesExt Plug-in Module	151
Configuration Parameters of CertificatePoliciesExt	152
CertificatePoliciesExt Rule	155
CertificateRenewalWindowExt Plug-in Module	156
Configuration Parameters of CertificateRenewalWindowExt	157
CertificateScopeOfUseExt Plug-in Module	161
Configuration Parameters of CertificateScopeOfUseExt	162
CRLDistributionPointsExt Plug-in Module	166
Configuration Parameters of CRLDistributionPointsExt	166
CRLDistributionPointsExt Rule	170
ExtendedKeyUsageExt Plug-in Module	171
Configuration Parameters of ExtendedKeyUsageExt	173
CODESigningExt Rule	175
OCSPSigningExt Rule	176
GenericASN1Ext Plug-in Module	177
Configuration Parameters of GenericASN1Ext	179
GenericASN1Ext Rule	184
IssuerAltNameExt Plug-in Module	184
Configuration Parameters of IssuerAltNameExt	185
KeyUsageExt Plug-in Module	189
Configuration Parameters of KeyUsageExt	191
CMCertKeyUsageExt Rule	196
RMCertKeyUsageExt Rule	197
ServerCertKeyUsageExt Rule	198
ClientCertKeyUsageExt Rule	199
ObjSignCertKeyUsageExt Rule	201

CRLSignCertKeyUsageExt	202
NameConstraintsExt Plug-in Module	202
Configuration Parameters of NameConstraintsExt	203
NameConstraintsExt Rule	210
NSCCommentExt Plug-in Module	211
Configuration Parameters of NSCCommentExt	212
NSCCommentExt Rule	214
NSCertTypeExt Plug-in Module	215
Configuration Parameters of NSCertTypeExt	218
NSCertTypeExt Rule	220
OCSPNoCheckExt Plug-in Module	220
Configuration Parameters of OCSPNoCheckExt	222
OCSPNoCheckExt Rule	223
PolicyConstraintsExt Plug-in Module	224
Configuration Parameters of PolicyConstraintsExt	224
PolicyConstraintsExt Rule	227
PolicyMappingsExt Plug-in Module	227
Configuration Parameters of PolicyMappingsExt	228
PolicyMappingsExt Rule	231
PrivateKeyUsagePeriodExt Plug-in Module	231
Configuration Parameters of PrivateKeyUsagePeriodExt	232
RemoveBasicConstraintsExt Plug-in Module	233
Configuration Parameters of RemoveBasicConstraintsExt	234
SubjectAltNameExt Plug-in Module	235
Configuration Parameters of SubjectAltNameExt	237
SubjectAltNameExt Rule	240
SubjectDirectoryAttributesExt Plug-in Module	241
Configuration Parameters of SubjectDirectoryAttributesExt	242
SubjectKeyIdentifierExt Plug-in Module	245
Configuration Parameters of SubjectKeyIdentifierExt	246
SubjectKeyIdentifierExt Rule	248
 Chapter 5 Mapper Plug-in Modules	 251
Overview of Mapper Modules	252
LdapCaSimpleMap Plug-in Module	255
Configuration Parameters of LdapCaSimpleMap	256
LdapCaCertMap Mapper	258
LdapCrlMap Mapper	259
LdapDNCompsMap Plug-in Module	259
Configuration Parameters of LdapDNCompsMap	262
LdapDNExactMap Plug-in Module	264
Configuration Parameters of LdapDNExactMap	265
LdapSimpleMap Plug-in Module	265

Configuration Parameters of LdapSimpleMap	266
LdapUserCertMap Mapper	267
LdapSubjAttrMap Plug-in Module	268
Configuration Parameters of LdapSubjAttrMap	268
Chapter 6 Publisher Plug-in Modules	271
Overview of Publisher Modules	272
FileBasedPublisher Plug-in Module	274
Configuration Parameters of FileBasedPublisher	274
LdapCaCertPublisher Plug-in Module	275
Configuration Parameters of LdapCaCertPublisher	276
LdapCaCertPublisher Publisher	277
LdapUserCertPublisher Plug-in Module	277
Configuration Parameters of LdapUserCertPublisher	278
LdapUserCertPublisher Publisher	279
LdapCrlPublisher Plug-in Module	279
Configuration Parameters of LdapCrlPublisher	280
LdapCrlPublisher Publisher	281
OCSPPublisher Plug-in Module	281
Configuration Parameters of OCSPPublisher	281
Chapter 7 CRL Extension Plug-in Modules	283
Overview of CRL Extension Modules	284
AuthorityKeyIdentifier Rule	285
CRLNumber Rule	287
CRLReason Rule	288
HoldInstruction Rule	290
InvalidityDate Rule	291
IssuerAlternativeName Rule	293
IssuingDistributionPoint Rule	297
Chapter 8 Log Plug-in Modules	301
Overview of Log Modules	301
file Plug-in Module	303
Configuration Parameters of file	304
Audit Log Event Listener	306
Error Log Event Listener	307
System Log Event Listener	308
NTEventLog Plug-in Module	308
Configuration Parameters of NTEventLog	309
NTAudit Event Listener	310
NTSystem Event Listener	310

Appendix A Distinguished Names	313
What Is a Distinguished Name?	313
Distinguished Name Components	314
Root Distinguished Name	315
Base Distinguished Name	315
DNs in Certificate Management System	316
Extending Attribute Support	318
Adding New or Proprietary Attributes	319
Adding Attributes to an Enrollment Form	320
Changing the DER Encoding Order	322
Role of Distinguished Names in Certificates	323
DNs in End-Entity Certificates	324
DNs in CA Certificates	324
Selecting DNs for Certificates	325
DN Patterns and Certificate Subject Names	325
 Appendix B Object Identifiers	 329
What's an Object Identifier?	329
Registration of Object Identifiers	330
 Appendix C Certificate and CRL Extensions	 331
Introduction to Certificate Extensions	331
Structure of Certificate Extensions	334
Sample Certificate Extensions	335
Recommendations for Certificate Extension Use	335
Standard X.509 v3 Certificate Extensions	341
authorityInfoAccess	343
authorityKeyIdentifier	344
basicConstraints	345
certificatePolicies	346
cRLDistributionPoints	347
extKeyUsage	348
issuerAltName	350
keyUsage	351
nameConstraints	354
OCSPNocheck	354
policyConstraints	355
policyMappings	356
privateKeyUsagePeriod	357
subjectAltName	358
subjectDirectoryAttributes	359
subjectKeyIdentifier	360
Introduction to CRL Extensions	361

Structure of CRL Extensions	361
Sample CRL and CRL Entry Extensions	363
Standard X.509 v3 CRL Extensions	364
Extensions for CRLs	364
authorityKeyIdentifier	364
CRLNumber	365
deltaCRLIndicator	365
issuerAltName	366
issuingDistributionPoint	366
CRL Entry Extensions	367
certificateIssuer	367
holdInstructionCode	368
invalidityDate	368
reasonCode	369
Netscape-Defined Certificate Extensions	369
netscape-cert-type	370
netscape-comment	371
CA Certificates and Extension Interactions	371
Index	373

About This Guide

The *CMS Plug-Ins Guide* provides reference information about all the plug-in modules provided with iPlanet Certificate Management Server (CMS). Plug-in modules help you configure and customize Certificate Management System, and use it for issuing and managing certificates to various end entities, such as web browsers (users), servers, Virtual Private Network (VPN) clients, and Cisco™ routers.

NOTE

Sun™ ONE Certificate Server was previously known as iPlanet™ Certificate Management System. The product was renamed shortly before the launch of this 4.7 release.

The late renaming of this product has resulted in a situation where the new product name is not fully integrated into the shipping product. In particular, you will see the product referenced as iPlanet Certificate Management Server (CMS) within the product GUI and within the product documentation. For this release, please consider iPlanet Certificate Management Server and Sun™ ONE Certificate Server as interchangeable names for the same product.

This chapter has the following sections:

- What's in This Guide (page 12)
- What You Should Already Know (page 13)
- Conventions Used in This Guide (page 14)
- Where to Go for Related Information (page 15)

What's in This Guide

This guide covers topics that are listed below. You should use this guide in conjunction with the other CMS documentation, such as the one that explains how to install and setup Certificate Management System. For a complete list of CMS documentation, see section “Where to Go for Related Information,” available later in this preface.

- “About This Guide” Describes what’s covered in this guide, what you should already know, and where to look for more information.
- Chapter 1, “Authentication Plug-in Modules” Describes the plug-in modules that you can use for authenticating end-users during certificate enrollment and it helps you decide on the authentication method suitable for your PKI setup.
- Chapter 2, “Job Plug-in Modules” Describes the plug-in modules that enable you to automate certain certificate-related tasks—such as notifying agents when a request gets queued, notifying users before their certificates expire, and removing expired certificates from the directory—to ease administration overheads.
- Chapter 3, “Constraints Policy Plug-in Modules” Describes the plug-in modules that you can use to govern the formulation of certificate content, such as key size, signing algorithm, validity period, and so on, and issuance of certificates.
- Chapter 4, “Certificate Extension Plug-in Modules” Describes the plug-in modules that enable you to add standard (X.509) and proprietary certificate extensions to certificate requests.
- Chapter 5, “Mapper Plug-in Modules” Describes the plug-in modules that enable you to configure a Certificate Manager to locate directory entries for publishing certificates and CRLs to the directory.
- Chapter 6, “Publisher Plug-in Modules” Describes the plug-in modules that enable you to configure a Certificate Manager to publish certificates to the correct attribute of the located directory entries.
- Chapter 7, “CRL Extension Plug-in Modules” Describes the plug-in modules that enable you to configure a Certificate Manager to set CRL extensions in CRLs before generating them and publishing them to a directory.
- Chapter 8, “Log Plug-in Modules” Describes the plug-in modules that enable you to configure CMS logs.

Appendixes

- Appendix A, “Distinguished Names” Briefly explains what are distinguished names (DNs) and describes how DN are used in Certificate Management System.
- Appendix B, “Object Identifiers” Briefly explains what are object identifiers (OIDs) and describes significance of registering OIDs.
- Appendix C, “Certificate and CRL Extensions” Summarizes the standard certificate and CRL extensions defined by X.509 version 3 and the extensions defined by Netscape before this version was finalized. Recommends extensions to use with specific kinds of certificates, including both PKIX Part 1 recommendations and Netscape extensions that must be supported to maintain compatibility with early versions of Netscape products.

What You Should Already Know

This guide is intended for experienced system administrators who are planning to deploy Certificate Management System. CMS agents should refer to *CMS Agent's Guide* for information on how to perform agent tasks, such as handling certificate requests and revoking certificates.

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
 - The 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://docs.sun.com/db?p=coll/S1_nsCMS_42_Resources

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

- Are familiar with the role of iPlanet Console in managing Netscape version 4.x servers. Otherwise, see the accompanying manual, *Managing Servers with iPlanet Console*.
- Are reading this guide in conjunction with the documentation listed in section “Where to Go for Related Information” on page 15.

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”—Indicates 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, which can be found in *CMS Installation and Setup Guide*.

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 or placeholders. 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://<hostname>:<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 iPlanet 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.

Where to Go for Related Information

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

- `<server_root>` is the directory where the CMS binaries are kept (which you specify during installation).

- `<instance_id>` is the ID for this instance of Certificate Management System (specified during installation).

The documentation set for Certificate Management System includes the following:

- *Managing Servers with iPlanet Console*

Provides background information on basic cryptography concepts and the role of iPlanet Console. To view the HTML version of this guide, open this file:

`<server_root>/manual/en/admin/help/contents.htm`

- *CMS Installation and Setup Guide*

Describes how to plan for, install, and administer Certificate Management System. To access the installation and configuration information from within the CMS Installation Wizard or from the CMS window (within iPlanet Console), click any help button.

To view the HTML version of this guide, open this file:

`<server_root>/manual/en/cert/setup_guide/contents.htm`

- *CMS Plug-Ins Guide*(this guide)

Provides detailed reference information on CMS plug-ins. To access this information from the CMS window within iPlanet Console, click any help button.

To view the HTML version of this guide, open this file:

`<server_root>/manual/en/cert/plugin_guide/contents.htm`

- *CMS Command-Line Tools Guide*

Provides detailed reference information on CMS tools.

To view the HTML version of this guide, open this file:

`<server_root>/manual/en/cert/tools_guide/contents.htm`

- *CMS Customization Guide*

Provides detailed reference information on customizing the HTML-based agent and end-entity interfaces.

To view the HTML version of this guide, open this file:

`<server_root>/manual/en/cert/custom_guide/contents.htm`

- *CMS Agent's Guide*

Provides detailed reference information on CMS agent interfaces. To access this information from the Agent Services pages, click any help button.

To view the HTML version of this guide, open this file:

```
<server_root>/cert-<instance_id>/web/agent/manual/agent_guide/
contents.htm
```

To view the PDF version of this guide, open this file:

```
<server_root>/manual/en/cert/pdf/cms42sp2agent.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.

To view the HTML version of this guide, open this file:

```
<server_root>/cert-<instance_id>/web/ee/manual/ee_guide/
contents.htm
```

NOTE	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.
-------------	---

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

For the latest information about Certificate Management System, including current release notes, technical notes, and deployment information, check this site:

```
http://docs.sun.com/?p=coll/S1_s1CertificateServer_47
```


Authentication Plug-in Modules

iPlanet Certificate Management Server (CMS) provides a set of authentication plug-in modules that enable you to configure a Certificate Manager or Registration Manager to authenticate end users, based on specified criteria, when they enroll for a certificate. This chapter explains the authentication modules that are installed with the Certificate Manager and Registration Manager—it lists and briefly describes the modules and then explains each one in detail.

The chapter has the following sections:

- Overview of Authentication Modules (page 20)
- Manual Authentication (page 23)
- UidPwdDirAuth Plug-in Module (page 24)
- UidPwdPinDirAuth Plug-in Module (page 30)
- NISAuth Plug-in Module (page 37)
- PortalEnroll Plug-in Module (page 44)
- SSOAuthentication Plug-In Module (page 52)
- Certificate-Based Enrollment (page 53)
- Enrollment Forms (page 57)

Overview of Authentication Modules

Certificate Management System supports both manual and automated certificate issuance.

- In the manual method of certificate issuance, end entities supply most of the information required by the server to formulate certificate requests and issue certificates. Manual issuance is also dependent on human agents; it requires that all end-entity certificate requests be approved by agents before the server can process the requests. To understand the role of an agent in your PKI, see section “Agents” in Chapter 13, *Managing Privileged Users and Groups* of *CMS Installation and Setup Guide*.
- In the automated method of certificate issuance, repositories, such as directories, supply part of the end-entity information. End entities only supply certain information—for example, a user ID and password—contained in the repository during certificate enrollment. The server uses this information for authenticating end entities before retrieving information required to formulate the certificate request from the repository.

For details on the manual method of certificate issuance, see “Manual Authentication” on page 23. For the automated method of certificate issuance, Certificate Management System provides a set of plug-in modules. Plug-in modules are implemented as Java classes and are registered in the CMS authentication framework. The Authentication Plugin Registration tab of the CMS window (see Figure 1-1) lists all the modules and the corresponding classes that are currently registered with a CMS instance.

Figure 1-1 Default authentication modules for end-user enrollment

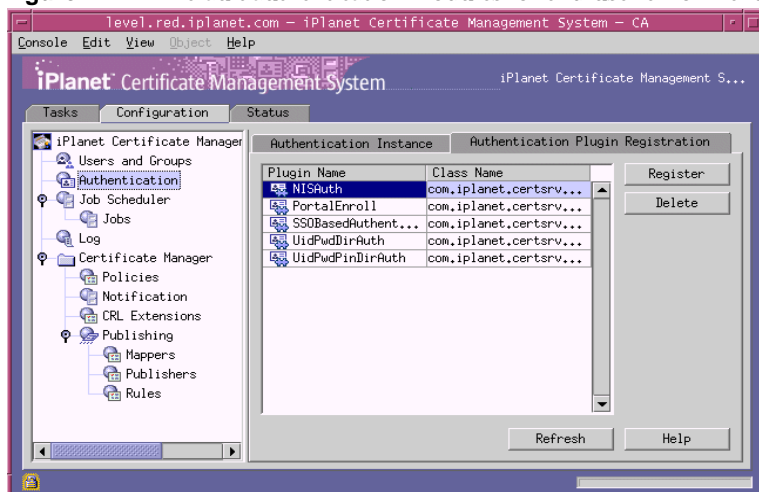


Table 1-1 lists the authentication modules provided for the Certificate Manager and Registration Manager; no authentication modules are provided for the Data Recovery Manager as it does not function as an enrollment authority in a PKI. You can use these modules to configure a Certificate Manager and Registration Manager to employ a specific authentication method during certificate enrollments.

Note that the name of the Java class for an authentication plug-in is in this format:

```
com.iplanet.certsrv.authentication.<plugin_name>
```

where <plugin_name> is the name of a plug-in module. For example, the Java class for the `UidPwdDirAuth` module would be:

```
com.iplanet.certsrv.authentication.UidPwdDirAuthentication
```

Table 1-1 Authentication plug-in modules for end user certificate enrollments

Plug-in module name	Function
<code>NISAuth</code>	Authenticates end users based on their user IDs and passwords stored in a NIS server. Optionally, uses an LDAP directory for formulating certificate subject names. For details, see “NISAuth Plug-in Module” on page 37.
<code>PortalEnroll</code>	Authenticates online service users based on their user IDs and passwords stored in an LDAP directory. Also registers new users for the online service. For details, see “PortalEnroll Plug-in Module” on page 44.
<code>UidPwdDirAuth</code>	Authenticates end users based on their user IDs and passwords stored in an LDAP directory. For details, see “UidPwdDirAuth Plug-in Module” on page 24.
<code>UidPwdPinDirAuth</code>	Authenticates end users based on their user IDs, passwords, and PINs stored in an LDAP directory. For details, see “UidPwdPinDirAuth Plug-in Module” on page 30.

Because large corporations typically store corporatewide user, group, and network-resource data in LDAP-compliant directories, the default authentication modules provided for automated certificate enrollment use an LDAP directory for authenticating users or for formulating certificate subject names, or for both. If you already have an LDAP-compliant directory, such as iPlanet Directory Server, with end-user data, you can use that directory for any of the purposes mentioned above. For example, if you have an NIS server and LDAP directory installations, you can use the NIS server for authenticating end users and the directory for formulating certificate subject names; end users will be required to provide only their NIS user IDs and passwords during enrollment.

If you don't have a directory deployed, you may use the Directory Server instance created at the time of CMS installation; in the documentation, this instance is identified as the Configuration Directory. For a demonstration on how to use this directory for issuing certificates to end users, see Chapter 3, "Default Demo Installation" of *CMS Installation and Setup Guide*.

If you determine that the default authentication modules do not meet your requirements, you can develop a custom authentication module using the CMS SDK, which is available in the form of Java Docs at this location:

```
<server_root>/cms_sdk/cms_jdk/javadocs
```

For general guidelines on developing custom authentication modules and adding them to the CMS authentication framework, check the tutorials on authentication. Be sure to take a look at the authentication-specific samples available at this location: `<server_root>/cms_sdk/cms_jdk/samples/authentication`

For instructions on how to configure a Certificate Manager and a Registration Manager to use one or more of the authentication methods, see section "Configuring Authentication for End-User Enrollment" in Chapter 15, "Setting Up End-User Authentication" of *CMS Installation and Setup Guide*.

Keep in mind that in an automated certificate management setup, the Certificate Manager and Registration Manager use the configured authentication methods only during certificate enrollment. During certificate renewal, the servers rely on end users SSL client certificate for automated renewal. For automated revocation, the users can use their SSL-client certificate or a challenge password. For more information, see sections "Authentication for End Users During Certificate Renewal" and "Authentication for End Users During Certificate Revocation" in Chapter 15, "Setting Up End-User Authentication" of *CMS Installation and Setup Guide*.

Certificate Management System also provides HTML forms-based interfaces for all the authentication methods it supports. Your end entities can use these forms for certificate enrollment. Explanation of each enrollment form, along with the corresponding authentication module, is covered in "Enrollment Forms" on page 57. Certificate renewal and revocation forms are covered as a part of those processes. For details on individual form elements in the enrollment, renewal, and revocation forms, see the online help available by clicking the Help buttons on the HTML forms. You can also customize these forms to suit to your organization's requirements. For customization information, see *CMS Customization Guide*.

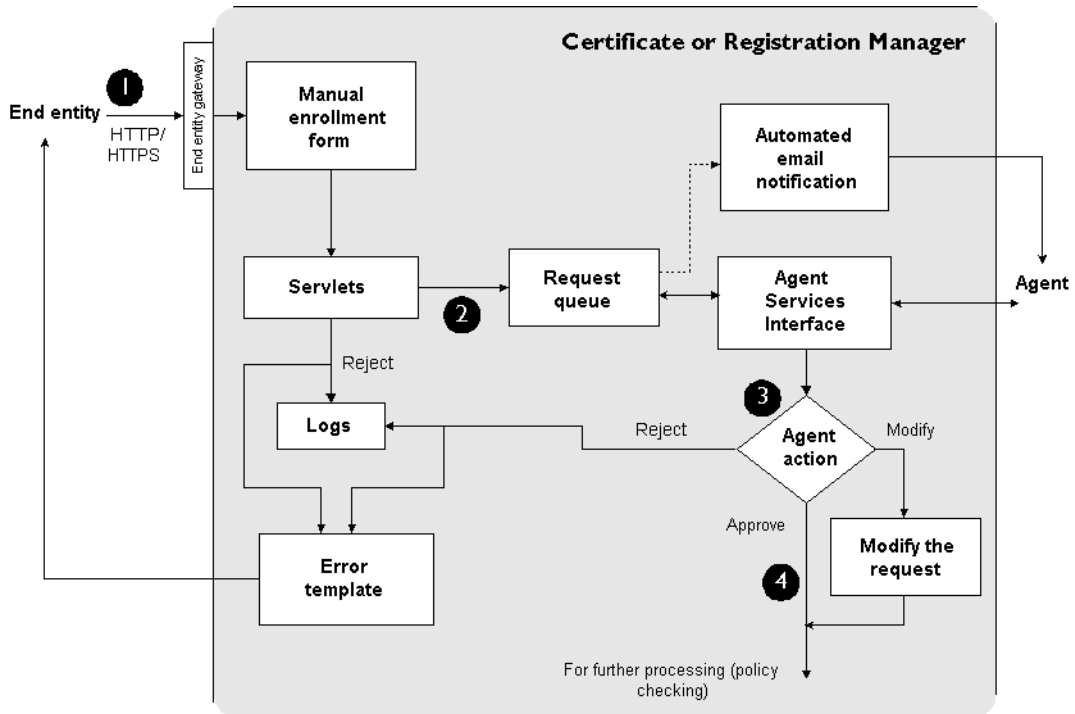
Manual Authentication

Manual authentication refers to operations which must be approved by a CMS agent, where no automated operation is possible. That is, a real person must log in to approve or reject request. By default, Certificate Management System provides manual-enrollment forms that enable you to request many types of certificates from the server. For details, see “Enrollment Forms” on page 57.

Note that the manual authentication method is hardcoded; you cannot configure it in any other way. This ensures that when the server receives requests that lack authentication credentials, it sends those requests to the request queue for agent approval. It also means that if you don’t configure a Certificate Manager or Registration Manager for any other authentication method, the server automatically sends all certificate-related requests to a queue where they await agent approval.

Figure 1-2 illustrates how the manual authentication method works during certificate enrollment.

Figure 1-2 Manual authentication of end entities during certificate enrollment



These are the steps shown in Figure 1-2:

1. In the manual enrollment form, the end entity enters the information needed to request 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 Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide* .
 - 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 certificate is delivered to the email address specified in the certificate request.

UidPwdDirAuth Plug-in Module

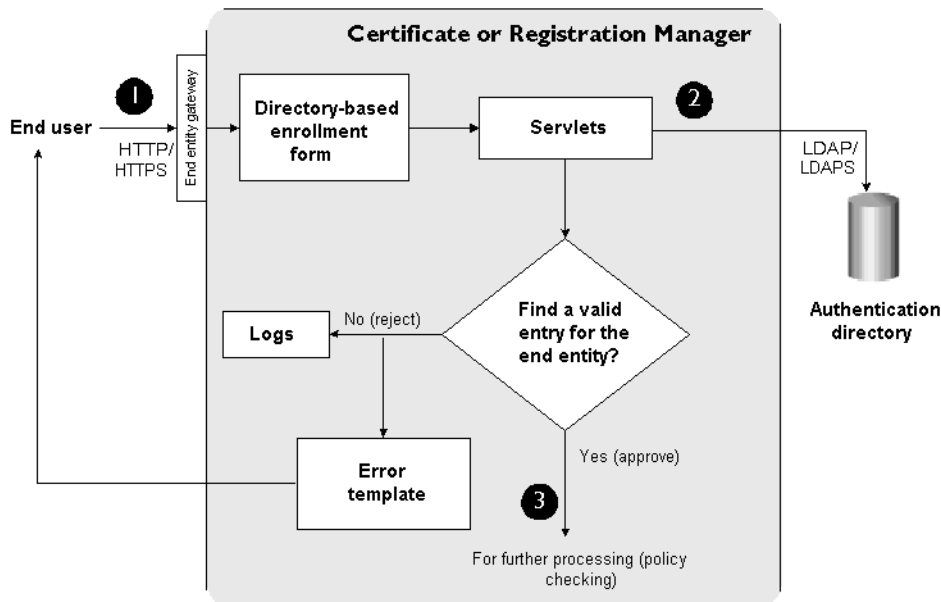
The `UidPwdDirAuth` plug-in module implements the directory-based authentication method. You can use this module for authenticating end users, provided their information is stored in an LDAP directory, during certificate enrollment.

Here’s how the enrollment method works: as part of configuring a Certificate Manager and a Registration Manager, or both, for authentication, you specify an LDAP directory that the server must use to authenticate end users. End users enroll for a certificate by entering their user IDs and passwords for this authentication

directory in an HTML form that is served by a Certificate Manager or Registration Manager (see “Enrollment Forms” on page 57). Once the server successfully authenticates an end user, it retrieves the rest of the information required to formulate the certificate from the directory.

Figure 1-3 illustrates how authentication based on a user ID and password works during certificate enrollment.

Figure 1-3 User ID- and password-based authentication of an end user



These are the steps shown in Figure 1-3:

1. In the directory-based certificate enrollment form, the end user enters a user ID and password for the directory and submits the request to a Certificate Manager or Registration Manager.
2. When the server receives the request, it looks up the directory that is configured for authenticating end users. The server verifies the authenticity of the user by checking the directory entries.
 - If the end user 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 user.

- If the end user has a valid entry in the directory, the server retrieves all the information required to construct the subject name for the user's certificate.

If, for some reason, the directory to which the server binds for authenticating the user ID and password is unavailable, the server returns an LDAP error code and writes it to the log. A sample log entry with an LDAP error code is shown below:

```
28/Jun/1999:18:40:25 -0700] conn=0 op=7 RESULT err=32 tag=101
nentries=0 etime=0]
```

3. Next, the server subjects the certificate request to policy processing. For details, see Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*.
 - 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 user a certificate.

The end user gets the certificate, which, if the server is configured to do so, is delivered to the email address specified in the request or in the directory. For information on configuring a Certificate Manager or Registration Manager to send automated notifications, see section “Notifications of Certificate Issuance to End Entities” in Chapter 16, “Setting Up Automated Notifications” of *CMS Installation and Setup Guide*.

Configuration Parameters of UIdPwDirAuth

In the configuration file, the UIdPwDirAuth module is identified as

```
auths.impl.UIdPwDirAuth.class=com.iplanet.certsrv.
authentication.UIdPwDirAuthentication.
```

In the CMS window, the module is identified as UIdPwDirAuth. Figure 1-4 shows how configurable parameters of the module are displayed in the CMS window.

Figure 1-4 Parameters defined in the UidPwdDirAuth module

The screenshot shows a window titled "Authentication Instance Editor". It contains the following fields and values:

Parameter	Value
Authentication Instance ID:	UserDirEnrollment
Authentication Plugin ID:	UidPwdDirAuth
dnpattern	E=\$attr.mail.1,CN=\$attr.cn,OU=\$dn.ou.2,O=\$dn.o,C=US
ldapStringAttributes	mail
ldapByteAttributes	
ldap.ldapconn.host	corpDirectory.siroe.com
ldap.ldapconn.port	389
ldap.ldapconn.secureConn	false
ldap.ldapconn.version	3
ldap.basedn	O=siroe.com
ldap.minConns	2
ldap.maxConns	8

At the bottom of the window are three buttons: "OK", "Cancel", and "Help".

Table 1-2 gives details about each of these parameters and their values.

Table 1-2 Description of parameters defined in the UidPwDirAuth module

Parameter	Description
dnpattern	<p>Specifies a string representing a subject name pattern to formulate from the directory attributes and entry DN.</p> <p>Permissible values: Any valid DN string composed from standard DN attributes, which must be separated by commas; see “DNs in Certificate Management System” on page 316.</p> <p>The syntax is illustrated in the following example:</p> <pre>E=\$attr.mail.1, CN=\$attr.cn, OU=\$dn.ou.2, O=\$dn.o, C=US</pre> <p>This sample configuration specifies that the subject name should be formulated as follows:</p> <ul style="list-style-type: none"> • E = the first mail LDAP attribute value in the user’s entry • CN = the (first) cn LDAP attribute value in the user’s entry • OU = the second ou value in the user’s entry DN • O = the (first) o value in the user’s entry DN • C = the string US <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 Netscape Communicator and other browsers. For Communicator, if you leave out E= in end-user 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>
ldapStringAttributes	<p>Specifies the list of LDAP string 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 formulate subject names for certificates or to make other policy decisions. For details, see “SubjectAltNameExt Plug-in Module” on page 235.</p> <p>Entering values for this parameter is optional.</p> <p>Permissible values: Any valid LDAP string attributes, separated by commas.</p> <p>Example: mail</p> <p>(This sample configuration specifies that the value of the mail attribute should be stored in the authentication token.)</p>

Table 1-2 Description of parameters defined in the UidPwdDirAuth module (*Continued*)

Parameter	Description
<code>ldapByteAttributes</code>	<p>Specifies the list of LDAP byte (binary) 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 for use by other modules—that is, values retrieved from this parameter can be used by policy modules to make certain policy decisions or to add additional information to users' certificates.</p> <p>For example, assume you have defined an LDAP binary attribute for storing users' pictures or fingerprints in your directory. You could develop a policy plug-in that adds users' pictures to their certificates as extensions.</p> <p>Entering values for this parameter is optional.</p> <p>Permissible values: Any valid LDAP byte attributes, separated by commas.</p> <p>Example: <code>jpegPhoto</code></p> <p>This sample configuration specifies that the value of the LDAP attribute named <code>jpegPhoto</code> (which is included in the standard <code>inetOrgPerson</code> object class) should be stored in the authentication token and be used to put the user's picture in his or her certificate.</p>
<code>ldap.ldapconn.host</code>	<p>Specifies the host name of the authentication directory.</p> <p>Permissible values: The name must be in the <code><machine_name>.<your_domain>.<domain></code> form.</p> <p>Example: <code>corpDirectory.siroe.com</code></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>Permissible values: Any valid port number.</p> <p>Example: <code>389</code></p>
<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"> • Check the box if the port is an SSL (HTTPS) port. If your authentication directory is configured for SSL-enabled communication (with or without SSL client authentication), choose this option. • Leave the box unchecked if the port is a non-SSL (HTTP) port. If your authentication directory is configured for basic authentication, choose this option (default).

Table 1-2 Description of parameters defined in the UIdPwDPinDirAuth module *(Continued)*

Parameter	Description
<code>ldap.ldapconn.version</code>	<p>Specifies the LDAP protocol version.</p> <p>Permissible values: 2 or 3.</p> <ul style="list-style-type: none"> 2 specifies LDAP version 2. If your authentication directory is based on Netscape Directory Server 1.x, choose 2. 3 specifies LDAP version 3. For Directory Server versions 3.x and later, choose 3. <p>Example: 3</p>
<code>ldap.basedn</code>	<p>Specifies the base DN for searching the authentication directory—the server uses the value of the <code>uid</code> field from the HTTP input (what a user enters in the enrollment form) and the base DN to construct an LDAP search filter.</p> <p>Permissible values: Any valid DN string of up to 255 characters.</p> <p>Example: <code>O=sirae.com</code></p>
<code>ldap.minConns</code>	<p>Specifies the minimum number of connections permitted to the authentication directory.</p> <p>Permissible values: 1 to 3.</p> <p>Example: 2</p>
<code>ldap.maxConns</code>	<p>Specifies the maximum number of connections permitted to the authentication directory.</p> <p>Permissible values: 3 to 10.</p> <p>Example: 8</p>

UIdPwDPinDirAuth Plug-in Module

The `UIdPwDPinDirAuth` plug-in module implements the directory- and PIN-based authentication method. You can use this module for authenticating users in the global LDAP domain during certificate enrollment. This authentication method is functionally very similar to the directory-based authentication explained in “UIdPwDPinDirAuth Plug-in Module” on page 24, except that for stronger authentication you combine a PIN or one-time password with the end users’ user IDs and passwords.

Here's how the enrollment method works: as a part of setting up a Certificate Manager or a Registration Manager, or both for end-user authentication, you specify the LDAP directory that the server must use to authenticate end users. End users enroll for a certificate by entering their user IDs, passwords, and PINs in an HTML form that is served by the Certificate Manager or Registration Manager (see "Enrollment Forms" on page 57). Once the server successfully authenticates an end user, it retrieves the rest of the information required to formulate the certificate from the directory. You can also configure the server to either retain or remove the PIN from the directory following successful authentication.

Normally, user entries in directories do not contain PINs. In order to use the `UidPwdPinDirAuth` module, you must first populate the directory that you intend to use for authentication with unique PINs for users; each user to whom you intend to issue a certificate must know his or her PIN at the time of certificate enrollment, as he or she will be required to enter it in the enrollment form. To aid you in the process of generating unique PINs for users and adding them to the directory, Certificate Management System provides a command-line tool called the PIN Generator. For information about this tool, see *CMS Command-Line Tools Guide*.

Configuration Parameters of UidPwdPinDirAuth

In the configuration file, the `UidPwdPinDirAuth` module is identified as `auths.impl.UidPwdPinDirAuth.class=com.ipplanet.certsrv.authentication.UidPwdPinDirAuthentication`.

In the CMS window, the module is identified as `UidPwdPinDirAuth`. Figure 1-5 shows how configurable parameters of the module are displayed in the CMS window.

Figure 1-5 Parameters defined in the UidPwdPinDirAuth module

The screenshot shows the 'Authentication Instance Editor' window. At the top, 'Authentication Instance ID' is 'PinDirEnrollment' and 'Authentication Plugin ID' is 'UidPwdPinDirAuth'. The main area contains several fields: 'removePin' is checked; 'pinAttr' is 'pin'; 'dnpattern' is 'E=\$attr.mail.1, CN=\$attr.cn, OU=\$dn.ou.2, O=\$dn.o, C=US'; 'ldapStringAttributes' is 'mail'; 'ldapByteAttributes' is empty; 'ldap.ldapconn.host' is 'corpDirectory.siroe.com'; 'ldap.ldapconn.port' is '389'; 'ldap.ldapconn.secureConn' is unchecked; 'ldap.ldapconn.version' is '3'; 'ldap.ldapauth.bindDN' is 'CN=pinmanager'; 'password' is masked with asterisks; 'ldap.ldapauth.clientCertNickname' is empty; 'ldap.ldapauth.authtype' is 'BasicAuth'; 'ldap.basedn' is 'O=siroe.com'; 'ldap.minConns' is '3'; and 'ldap.maxConns' is '9'. At the bottom, there is a template for cert Subject Name and 'OK', 'Cancel', and 'Help' buttons.

Table 1-3 gives details about each of these parameters.

Table 1-3 Description of parameters defined in the UidPwdPinDirAuth module

Parameter	Description
removePin	<p>Specifies whether to remove PINs from the authentication directory (after end users successfully authenticate). Removing PINs from the directory restricts users from enrolling more than once, and thus prevents them from getting more than one certificate.</p> <ul style="list-style-type: none">• Check the box if you want the server to remove PINs from the directory after successful authentication. If you set the value to <code>true</code>, you must also specify the values for the <code>ldap.ldapauth.bindDN</code> and <code>password</code> parameters.• Uncheck the box if you want the server to leave PINs in the directory after authentication.

Table 1-3 Description of parameters defined in the UidPwdPinDirAuth module (*Continued*)

Parameter	Description
<code>pinAttr</code>	<p>Specifies the authentication directory attribute for PINs. If you used the <i>PIN Generator</i> utility (provided with Certificate Management System), 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 section “Arguments” in Chapter 4, “PIN Generator Tool” of <i>CMS Command-Line Tools Guide</i>.</p> <p>Permissible values: Any valid attribute name.</p> <p>Example: <code>pin</code></p>
<code>dnpattern</code>	<p>Specifies a string representing a subject name pattern to formulate from the directory attributes and entry DN.</p> <p>Permissible values: Any valid DN string composed from standard DN attributes, which must be separated by commas; see “DNs in Certificate Management System” on page 316.</p> <p>The syntax is illustrated in the following example:</p> <pre>E=\$attr.mail.1, CN=\$attr.cn, OU=\$dn.ou.2, O=\$dn.o, C=US</pre> <p>This sample configuration specifies that the subject name should be formulated as follows:</p> <ul style="list-style-type: none"> • <code>E</code> = the first <code>mail</code> LDAP attribute value in the user’s entry • <code>CN</code> = the (first) <code>cn</code> LDAP attribute value in the user’s entry • <code>OU</code> = the second <code>ou</code> value in the user’s entry DN • <code>O</code> = the (first) <code>o</code> value in the user’s entry DN • <code>C</code> = the string <code>US</code> <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 Netscape Communicator and other browsers. For Communicator, if you leave out <code>E=</code> in end-user 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>

Table 1-3 Description of parameters defined in the UidPwdPinDirAuth module (*Continued*)

Parameter	Description
<code>ldapStringAttributes</code>	<p>Specifies the list of LDAP string 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 formulate subject names for certificates or to make other policy decisions. For details, see “SubjectAltNameExt Plug-in Module” on page 235.</p> <p>Entering values for this parameter is optional.</p> <p>Permissible values: Any valid LDAP string attributes, separated by commas.</p> <p>Example: <code>mail</code></p> <p>(This sample configuration specifies that the value of the <code>mail</code> attribute should be stored in the authentication token.)</p>
<code>ldapByteAttributes</code>	<p>Specifies the list of LDAP byte (binary) 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 for use by other modules—that is, values retrieved from this parameter can be used by policy modules to make certain policy decisions or to add additional information to users’ certificates.</p> <p>For example, assume you have defined an LDAP binary attribute for storing users’ pictures or fingerprints in your directory. You could develop a policy plug-in that adds users’ pictures to their certificates as extensions.</p> <p>Entering values for this parameter is optional.</p> <p>Permissible values: Any valid LDAP byte attributes, separated by commas.</p> <p>Example: <code>jpegPhoto</code></p> <p>This sample configuration specifies that the value of the LDAP attribute named <code>jpegPhoto</code> (which is included in the standard <code>inetOrgPerson</code> object class) should be stored in the authentication token and be used to put the user’s picture in his or her certificate.</p>
<code>ldap.ldapconn.host</code>	<p>Specifies the host name of the authentication directory.</p> <p>Permissible values: The name must be in the <code><machine_name>.<your_domain>.<domain></code> form.</p> <p>Example: <code>corpDirectory.siroe.com</code></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>Permissible values: Any valid port number.</p> <p>Example: <code>389</code></p>

Table 1-3 Description of parameters defined in the UidPwdPinDirAuth module (*Continued*)

Parameter	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"> • Check the box if the port is an SSL (HTTPS) port. If your authentication directory is configured for SSL-enabled communication (with or without SSL client authentication), choose this option. • Leave the box unchecked if the port is a non-SSL (HTTP) port. If your authentication directory is configured for basic authentication, choose this option (default).
<code>ldap.ldapconn.version</code>	<p>Specifies the LDAP protocol version.</p> <p>Permissible values: 2 or 3.</p> <ul style="list-style-type: none"> • 2 specifies LDAP version 2. If your authentication directory is based on Netscape Directory Server 1.x, choose 2. • 3 specifies LDAP version 3. For Directory Server versions 3.x and later, choose 3 (default). <p>Example: 3</p>
<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 selected <code>removePin</code>. 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>Permissible values: A valid bind DN.</p> <p>Example: <code>CN=pinmanager</code></p>
<code>password</code>	<p>Specifies the password associated with the DN specified by the <code>ldap.ldapauthbindDN</code> parameter. when you save your changes, the server stores the password in the single sign-on password cache and uses it for subsequent start ups (see section "Required Start-up Information" in Chapter 8, "Starting and Stopping CMS Instances" of <i>CMS Installation and Setup Guide</i>.)</p> <p>You need to specify this parameter only if you've selected <code>removePin</code>.</p>

Table 1-3 Description of parameters defined in the UIdPwdPinDirAuth module *(Continued)*

Parameter	Description
<code>ldap.ldapauth.clientCertNickname</code>	<p>Specifies the nickname or the friendly name of the certificate to be used for SSL client authentication to the authentication directory in order to remove PINs. Make sure that the certificate is valid and has been signed by a CA that is trusted in the authentication directory's certificate database, and that the authentication directory's <code>certmap.conf</code> file has been configured to correctly map the certificate to a DN in the directory. (This is needed for PIN removal only.)</p> <p>Permissible values: Enter the name of a currently valid CMS certificate, for example, its SSL server certificate.</p> <p>Example: <code>Server-Cert</code></p>
<code>ldap.ldapauth.authtype</code>	<p>Specifies the authentication type—basic authentication or SSL client authentication—required in order to remove PINs from the authentication directory.</p> <p>Permissible values: <code>BasicAuth</code> or <code>SslClientAuth</code>.</p> <ul style="list-style-type: none"> <code>BasicAuth</code> specifies basic authentication. If you choose this option, be sure to enter the correct values for <code>ldap.ldapauth.bindDN</code> and <code>password</code> parameters; the server uses the DN from the <code>ldap.ldapauth.bindDN</code> attribute to bind to the directory (default). <code>SslClientAuth</code> specifies SSL client authentication. If you choose this option, be sure to set the value of the <code>ldap.ldapconn.secureConn</code> parameter to <code>true</code> and the value of the <code>ldap.ldapauth.clientCertNickname</code> parameter to the nickname of the certificate to be used for SSL client authentication. <p>Example: <code>BasicAuth</code></p>
<code>ldap.basedn</code>	<p>Specifies the base DN for searching the authentication directory—the server uses the value of the <code>uid</code> field from the HTTP input (what a user enters in the enrollment form) and the base DN to construct an LDAP search filter.</p> <p>Permissible values: Any valid DN string of up to 255 characters.</p> <p>Example: <code>O=siroe.com</code></p>
<code>ldap.minConns</code>	<p>Specifies the minimum number of connections permitted to the authentication directory.</p> <p>Permissible values: 1 to 3.</p> <p>Example: 3</p>

Table 1-3 Description of parameters defined in the UidPwdPinDirAuth module (*Continued*)

Parameter	Description
<code>ldap.maxConns</code>	Specifies the maximum number of connections permitted to the authentication directory. Permissible values: 3 to 10. Example: 9

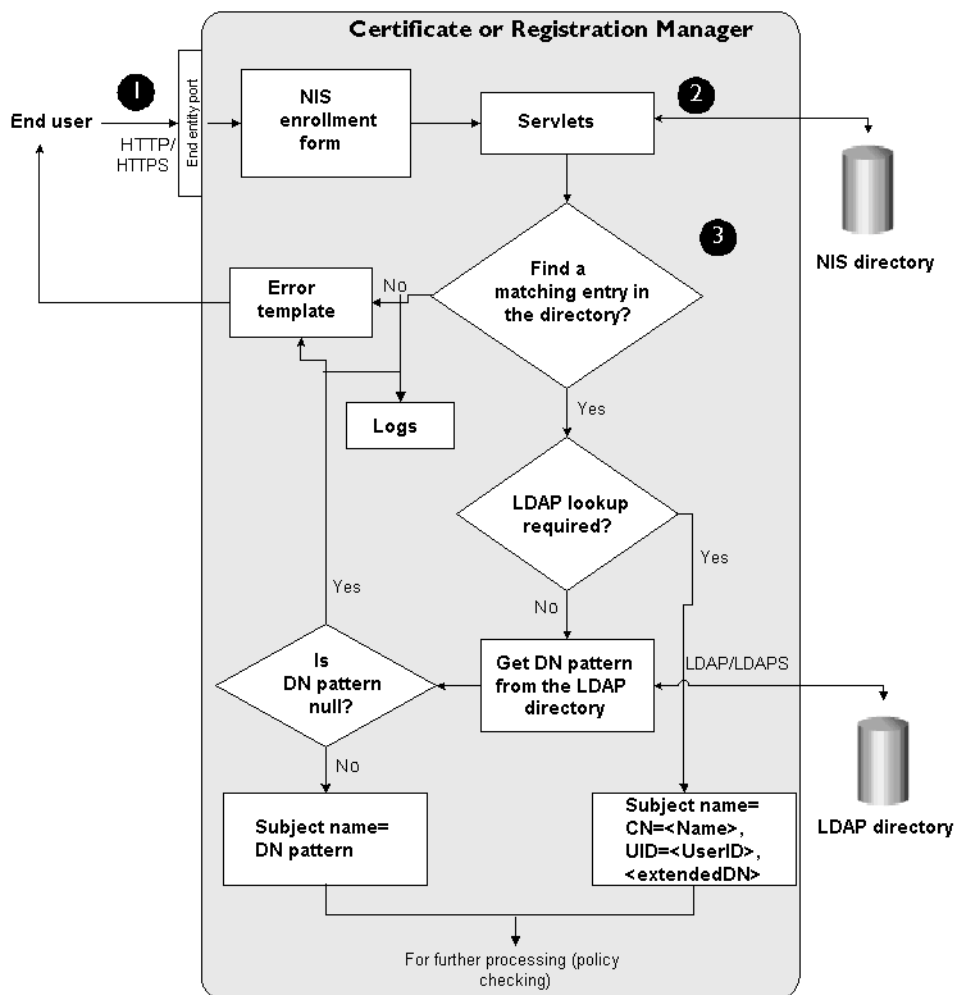
NISAuth Plug-in Module

The `NISAuth` module implements the NIS server-based authentication. You can use the module for authenticating unprivileged users in the NIS domain during certificate enrollment. The module enables you to deploy Public Key Infrastructure (PKI) leveraging an existing NIS server installation—it enables you to configure a Certificate Manager or Registration Manager to authenticate end users, based on their user IDs and passwords stored in an existing NIS server, and to issue certificates.

Optionally, you can configure the authentication module to do an *LDAP correlation*—that is, use the NIS directory to authenticate users based on the user ID and password they enter in the enrollment form, but compose certificate subject names from an LDAP-compliant directory, such as iPlanet Directory Server. When using an LDAP directory to compose subject names, you can configure the module to search for and retrieve specific LDAP attribute values from the directory. The ability of the module to use an LDAP directory to form certificate subject names is useful in cases where the NIS server only stores user IDs and passwords and you don't want to formulate subject names using just common names and user IDs.

In the absence of an LDAP directory, subject names of all certificates issued by the server will be of the form `CN=<FirstName LastName>,UID=<UserID>`, where `First Name` and `Last Name` is a user's first and last names as specified in the NIS directory, and `UserID` is the user's NIS ID. To accommodate scenarios where the default subject-name form isn't adequate, the module supports a parameter named `extendedDN`. This parameter enables you to specify a suffix that the server should use for extending the default subject DN pattern.

Figure 1-6 illustrates how the NIS authentication module works during certificate enrollment.

Figure 1-6 NIS server-based authentication of an end user

These are the steps shown in Figure 1-6:

1. In the NIS server-based certificate enrollment form, the end user enters his or her user ID and password for the NIS server and submits the request to a Certificate Manager or Registration Manager.
2. When the server receives the request, it looks up the NIS server that is configured for authenticating end users. The server verifies the authenticity of the end user by checking the entries.

- If the end user does not have a valid entry in the NIS server, the Certificate Manager or Registration Manager rejects the request, logs an error message, and sends a rejection notification to the user.
- If the end user has a valid entry in the NIS server, the Certificate Manager or Registration Manager checks to see if any LDAP directory has been configured for retrieving attributes for constructing the certificate subject name. If a directory is specified, the server checks it for the user's entry, retrieves all the information required to construct the subject name, and adds the subject name to the certificate request. If a directory is unspecified, the server uses the NIS user's name, user ID, and extended DN (if specified) for the subject name.

If, for some reason, the directory to which the server binds for retrieving user attributes is unavailable, the server writes the appropriate LDAP error code to the log. A sample log entry with an LDAP error code is shown below:

```
30/Dec/1999:18:40:25 -0700] conn=0 op=7 RESULT err=32 tag=101
nentries=0 etime=0]
```

3. Next, the server subjects the certificate request to policy processing. For details, see Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*.
 - 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 user.
 - If the request passes all the configured policies, the server issues the end user a certificate.

The end user gets the certificate, which, if the server is configured to do so, is delivered to the email address specified in the request or in the directory; for information on configuring a Certificate Manager or Registration Manager to send automated notifications, see section “Notifications of Certificate Issuance to End Entities” in Chapter 16, “Setting Up Automated Notifications” of *CMS Installation and Setup Guide*.

Configuration Parameters of NISAuth

In the configuration file, the NISAuth module is identified as `auths.impl.NISAuth.class=com.ipplanet.certsrv.authentication.NISAuth`.

In the CMS window, the module is identified as NISAuth. Figure 1-7 shows how configurable parameters of the module are displayed in the CMS window.

Figure 1-7 Parameters defined in the NISAuth module

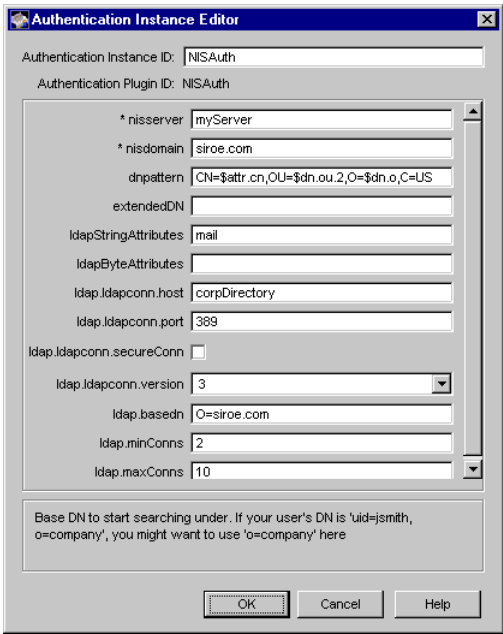


Table 1-4 gives details about each of these parameters and their values.

Table 1-4 Description of parameters defined in the NISAuth module

Parameter	Description
nisserver	Specifies the NIS server name. (In Unix, use the <code>ypwhich</code> command to find the NIS server name.) Permissible values: A valid server name. Example: <code>myServer</code>
nisdomain	Specifies the NIS domain name. (In Unix, use the <code>domainname</code> command to find the domain name.) Permissible values: A valid domain name. Example: <code>siroe.com</code>

Table 1-4 Description of parameters defined in the NISAuth module *(Continued)*

Parameter	Description
dnpattern	<p>Specifies a string representing a subject name pattern to formulate from the directory attributes and entry DN.</p> <p>Permissible values: Any valid DN string composed from standard DN attributes, which must be separated by commas; see “DNs in Certificate Management System” on page 316.</p> <p>The syntax is illustrated in the following example:</p> <pre>E=\$attr.mail.1, CN=\$attr.cn, OU=\$dn.ou.2, O=\$dn.o, C=US</pre> <p>This sample configuration specifies that the subject name should be formulated as follows:</p> <ul style="list-style-type: none"> • E = the first mail LDAP attribute value in the user’s entry • CN = the (first) cn LDAP attribute value in the user’s entry • OU = the second ou value in the user’s entry DN • O = the (first) o value in the user’s entry DN • C = the string US <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 Netscape Communicator and other browsers. For Communicator, if you leave out <code>E=</code> in end-user 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>
extendedDN	<p>Specifies the suffix that the server should use for extending the default subject DN when an LDAP directory for retrieving such information is not specified. The value you specify in this field is used by the sever to suffix the default subject name in certificates, which is in the form <code>CN=<FirstName LastName>,UID=<UserID>.</code></p> <p>Example: If you assign <code>OU=People,O=siroe.org,C=US</code> as the extended DN, subject names in certificates would be of this form:</p> <pre>CN=<FirstName LastName>,UID=<UserID>,OU=People, O=siroe.org,C=US</pre>

Table 1-4 Description of parameters defined in the NISAuth module (*Continued*)

Parameter	Description
<code>ldapStringAttributes</code>	<p>Specifies the list of LDAP string 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 formulate subject names for certificates or to make other policy decisions. For details, see “SubjectAltNameExt Plug-in Module” on page 235.</p> <p>Entering values for this parameter is optional.</p> <p>Permissible values: Any valid LDAP string attributes, separated by commas.</p> <p>Example: <code>mail</code></p> <p>(This sample configuration specifies that the value of the <code>mail</code> attribute should be stored in the authentication token.)</p>
<code>ldapByteAttributes</code>	<p>Specifies the list of LDAP byte (binary) attributes that should be considered <i>authentic</i> for the end user. If specified, the values corresponding to these attributes will be copied from the LDAP directory into the authentication token for use by other modules—that is, values retrieved from this parameter can be used by policy modules to make certain policy decisions or to add additional information to users’ certificates.</p> <p>For example, assume you have defined an LDAP binary attribute for storing users’ pictures or fingerprints in your directory. You could develop a policy plug-in that adds users’ pictures to their certificates as extensions.</p> <p>Entering values for this parameter is optional.</p> <p>Permissible values: Any valid LDAP byte attributes, separated by commas.</p> <p>Example: <code>jpegPhoto</code></p> <p>This sample configuration specifies that the value of the LDAP attribute named <code>jpegPhoto</code> (which is included in the standard <code>inetOrgPerson</code> object class) should be stored in the authentication token and be used to put the user’s picture in his or her certificate.</p>
<code>ldap.ldapconn.host</code>	<p>Specifies the host name of the LDAP directory.</p> <p>Permissible values: The name must be in the <code><machine_name>.<your_domain>.<domain></code> form.</p> <p>Example: <code>corpDirectory.siroe.com</code></p>
<code>ldap.ldapconn.port</code>	<p>Specifies the TCP/IP port at which the LDAP directory listens to requests from Certificate Management System.</p> <p>Permissible values: Any valid port number.</p> <p>Example: <code>389</code></p>

Table 1-4 Description of parameters defined in the NISAuth module *(Continued)*

Parameter	Description
<code>ldap.ldapconn.secureConn</code>	<p>Specifies the type—SSL or non-SSL—of the port at which the LDAP directory listens to requests from Certificate Management System.</p> <p>Permissible values: <code>true</code> or <code>false</code>.</p> <ul style="list-style-type: none"> <code>true</code> specifies that the port is an SSL (HTTPS) port. If your directory is configured for SSL-enabled communication (with or without SSL client authentication), choose this option. <code>false</code> specifies that the port is a non-SSL (HTTP) port. If your directory is configured for basic authentication, choose this option. <p>Example: <code>false</code></p>
<code>ldap.ldapconn.version</code>	<p>Specifies the LDAP protocol version of the LDAP directory.</p> <p>Permissible values: <code>true</code> or <code>false</code>.</p> <ul style="list-style-type: none"> <code>2</code> specifies LDAP version 2. If your directory is based on Netscape Directory Server 1.x, choose 2. <code>3</code> specifies LDAP version 3. For Directory Server versions 3.x and later, choose 3. <p>Example: <code>3</code></p>
<code>ldap.basedn</code>	<p>Specifies the base DN for searching the LDAP directory—the server uses the value of the <code>uid</code> field from the HTTP input (what a user enters in the enrollment form) and the base DN to construct an LDAP search filter.</p> <p>Permissible values: Any valid DN string of up to 255 characters.</p> <p>Example: <code>O=siroe.com</code></p>
<code>ldap.minConns</code>	<p>Specifies the minimum number of connections permitted to the LDAP directory.</p> <p>Permissible values: 1 to 3</p> <p>Example: <code>2</code></p>
<code>ldap.maxConns</code>	<p>Specifies the maximum number of connections permitted to the LDAP directory.</p> <p>Permissible values: 3 to 10</p> <p>Example: <code>10</code></p>

PortalEnroll Plug-in Module

The `PortalEnroll` module implements portal enrollment. This module enables you to issue certificates and create directory entries for users who do not yet have an entry in the directory. For example, if your company runs a portal service, such as `mysun.sun.com`, you can use the `PortalEnroll` module to issue certificates to new users when they register for the online service. You can also use the module to authenticate and issue certificates to your extranet users. For example, if you have deployed extranets for partners and vendors, you can use the module to authenticate and issue certificates to these users when they register for the service.

The `PortalEnroll` module does following:

- Performs dual operations, registration and authentication, eliminating the need for users to use separate forms to register for an online service and to request a certificate; the module enables deployment of certificates along with registration in an LDAP-compliant directory.
- Verifies the uniqueness of the new user's chosen user name against an LDAP-compliant user directory and uses the user name as the only authentication token required to obtain a certificate.
- Uses the information from the enrollment form to create new user entries and update directory entry attributes for unique usernames.
- Leverages an existing LDAP-compliant user directory, typically used for storing user information.

There are many advantages in issuing certificates to your user community:

- Certificates enable you to uniquely identify users and establish a relationship with users in that you can use their identities to track services and features utilized by these users and use this information to offer customized services to them—certificates become equivalent to the way online services utilize cookies for personalization.
- Certificates also enable you to make your online service subscription based—because a certificate's life is tied to its validity period, by issuing certificates with specific validity period you can enforce users to subscribe to your online service by renewing their certificate before its expiry.
- Certificates also enable you to remove people from your user base and add them back after giving them a credential—by making a certificate issued to a new user expire after a specific validity period you can restrict that user from using your service, and put the user back on service by forcing the user to renew the expired certificate after giving them a credential. For example, assume you have an extranet deployed for your partners. You have no prior

knowledge of people who will register as your partners, but you want them to register and you want to trust the information they provide during registration. By issuing them a certificate with a short validity period you can limit them from using your service for that period. In the meantime, you can verify their registration data and decide whether to allow them to continue using your service; if you want them to be your partners, you allow them to renew their certificates before they expire; if you don't want them as your partners, you reject their certificate renewal requests.

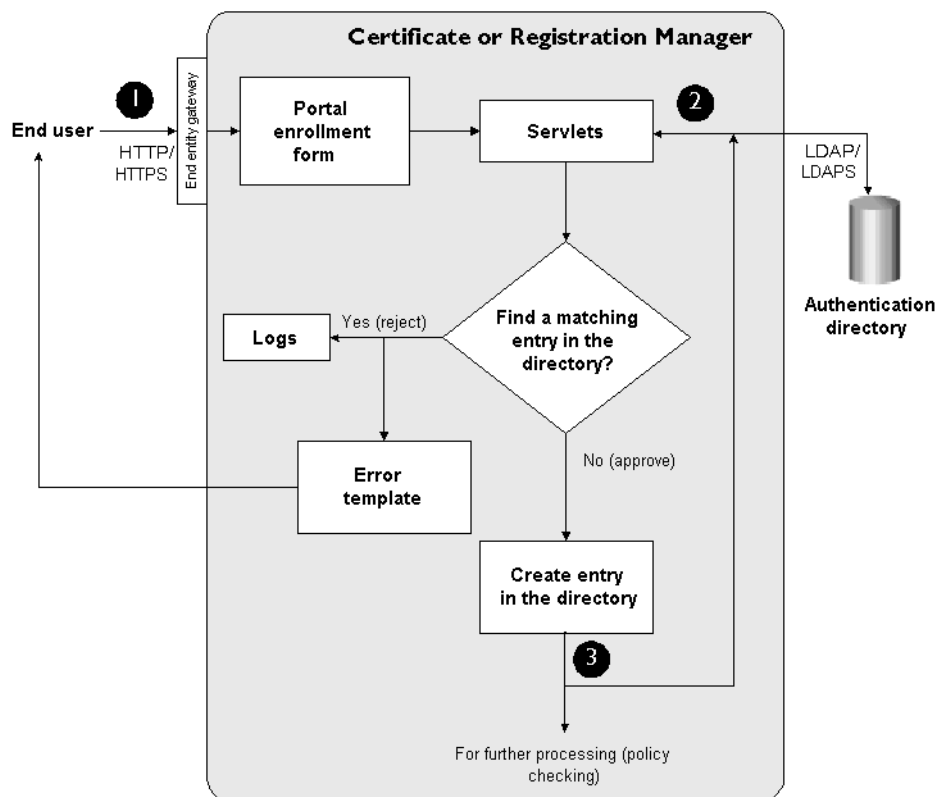
Note that Certificate Management System can send automated renewal notifications to users before their certificates expire; see “RenewalNotificationJob Plug-in Module” on page 69.

Functionally, the portal authentication module is very similar to the directory-based authentication module (see “UidPwdDirAuth Plug-in Module” on page 24) except that instead of binding to the directory as the enrolling user, Certificate Management System binds as some directory account with permission to create and update user entries. The server then queries the directory for the user name specified by the user and if it doesn't find a match, it adds the entry with all the standard LDAP field names that match the directory attributes.

For example, if the HTTP form input contains data such as surname, common name, and phone number, the corresponding LDAP attributes would be set in the directory; for details, see “Enrollment Forms” on page 57. The server also uses a combination of these attributes (which you can specify using the `dnpattern` parameter defined in the module) to construct subject names for certificates.

Note that the portal authentication module by default uses the standard LDAP object class named `inetOrgPerson` to create and update user entries. The input fields defined in the default portal enrollment form correspond to the attributes defined in this object class as defined in iPlanet Directory Server 4.x. The module is capable of reading and writing these attributes only. However, you can customize the module to accommodate all the fields supported by popular portals by extending the directory schema to include a new object class; you'll also be required to update the enrollment form to include attributes corresponding to the new object class. For guidelines on how to customize the module, check the sample located here: `<server_root>/cms_sdk/cms_jdk/samples/authentication`

Figure 1-8 illustrates how the portal authentication module works during certificate enrollment.

Figure 1-8 Portal authentication of an end user

These are the steps shown in Figure 1-8:

1. In the portal enrollment form, the end user enters registration information, such as a user name or ID, password, first name, last name, and mailing address, and submits the request to the server.
2. When the server receives the request, it verifies that the required fields contain appropriate information, for example, the values entered in the Password and Confirm Password fields match. Next, the server looks up the directory that is configured for authenticating portal service users for a matching user name.
 - o If the server finds a matching user name in the directory, it rejects the request, logs an error message, and sends a rejection notification to the end user.

- If the server fails to find a matching user name in the directory, it uses the registration information to create a user entry for the new user and add relevant attributes. The server also retrieves information required to construct the subject name for the certificate.

If, for some reason, the directory to which the server binds for authenticating the user ID and password is unavailable, the server returns an LDAP error code and writes it to the log. A sample log entry with an LDAP error code is shown below:

```
28/Jun/1999:18:40:25 -0700] conn=0 op=7 RESULT err=32 tag=101
nentries=0 etime=0]
```

3. Next, the server subjects the certificate request to policy processing. For details, see Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*.
 - 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 user. Note that if this happens, the user won’t be able to reregister using the same user name.
 - If the request passes all the configured policies, the server issues the end user a certificate.

The end user gets the certificate, which, if the server is configured to do so, is delivered to the email address specified in the request or in the directory; for information on configuring a Certificate Manager or Registration Manager to send automated notifications, see section “Notifications of Certificate Issuance to End Entities” in Chapter 16, “Setting Up Automated Notifications” of *CMS Installation and Setup Guide*.

Configuration Parameters of PortalAuth

In the configuration file, the `PortalEnroll` module is identified as `auths.impl.PortalEnroll.class=com.ipланet.certsrv.authentication.PortalEnroll`.

In the CMS window, the module is identified as `PortalEnroll`. Figure 1-9 shows how configurable parameters for the module are displayed in the CMS window.

Figure 1-9 Parameters defined in the PortalEnroll module

Authentication Instance Editor

Authentication Instance ID: PortalEnrollment

Authentication Plugin ID: PortalEnroll

dnpattern

E=\$attr.mail.1, CN=\$attr.cn, OU=\$dn.ou.2, O=\$dn.o, C=US

* ldap.ldapconn.host

portalDirectory.siroe.com

* ldap.ldapconn.port

389

ldap.ldapconn.secureConn

☐

ldap.ldapconn.version

3

* ldap.ldapauth.bindDN

CN=portalmanager

password

ldap.ldapauth.clientCertNickname

ldap.ldapauth.authtype

BasicAuth

* ldap.basedn

O=siroe.com

* ldap.objectclass

inetOrgPerson

ldap.minConns

3

ldap.maxConns

9

How to bind to the directory (for pin removal only)

OK

Cancel

Help

Table 1-5 gives details about each of these parameters and their values.

Table 1-5 Description of parameters defined in the PortalEnroll module

Parameter	Description
<code>dnpattern</code>	<p>Specifies a string representing a subject name pattern to formulate from the directory attributes and entry DN.</p> <p>Permissible values: Any valid DN string composed from standard DN attributes, which must be separated by commas; see “DNs in Certificate Management System” on page 316.</p> <p>The syntax is illustrated in the following example:</p> <pre>E=\$attr.mail.1, CN=\$attr.cn, OU=\$dn.ou.2, O=\$dn.o, C=US</pre> <p>This sample configuration specifies that the subject name should be formulated as follows:</p> <ul style="list-style-type: none"> • E = the first mail LDAP attribute value in the user’s entry • CN = the (first) cn LDAP attribute value in the user’s entry • OU = the second ou value in the user’s entry DN • O = the (first) o value in the user’s entry DN • C = the string US <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 Netscape Communicator and other browsers. For Communicator, if you leave out <code>E=</code> in end-user 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>
<code>ldap.ldapconn.host</code>	<p>Specifies the host name of the portal directory.</p> <p>Permissible values: The name must be in the <code><machine_name>.<your_domain>.<domain></code> form.</p> <p>Example: <code>portalDirectory.siroe.com</code></p>
<code>ldap.ldapconn.port</code>	<p>Specifies the TCP/IP port at which the portal directory listens to requests from Certificate Management System.</p> <p>Permissible values: Any valid port number.</p> <p>Example: 389</p>

Table 1-5 Description of parameters defined in the PortalEnroll module *(Continued)*

Parameter	Description
<code>ldap.ldapconn.secureConn</code>	<p>Specifies the type—SSL or non-SSL—of the port at which the portal directory listens to requests from Certificate Management System.</p> <ul style="list-style-type: none"> Check the box if the port is an SSL (HTTPS) port. If your portal directory is configured for SSL-enabled communication (with or without SSL client authentication), choose this option. Leave the box unchecked if the port is a non-SSL (HTTP) port. If your portal directory is configured for basic authentication, choose this option (default).
<code>ldap.ldapconn.version</code>	<p>Specifies the LDAP protocol version.</p> <p>Permissible values: 2 or 3.</p> <ul style="list-style-type: none"> 2 specifies LDAP version 2. If your portal directory is based on Netscape Directory Server 1.x, choose 2. 3 specifies LDAP version 3. For Directory Server versions 3.x and later, choose 3 (default). <p>Example: 3</p>
<code>ldap.ldapauth.bindDN</code>	<p>Specifies the user account to bind as in order to create and update user entries in the portal directory. It is recommended that you create and use a separate user account that has permission to create user entries and modify user attributes 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>Permissible values: A valid bind DN.</p> <p>Example: CN=Portal Registration Manager</p>
<code>password</code>	<p>Specifies the password associated with the DN specified by the <code>ldap.ldapauthbindDN</code> parameter.</p> <p>Permissible values: As applicable.</p>
<code>ldap.ldapauth.clientCertNickname</code>	<p>Specifies the nickname or the friendly name of the certificate to be used for SSL client authentication to the portal directory. Make sure that the certificate is valid and has been signed by a CA that is trusted in the portal directory's certificate database, and that the portal directory's <code>certmap.conf</code> file has been configured to correctly map the certificate to a DN in the directory.</p> <p>Permissible values: Enter the name of a currently valid CMS certificate, for example, its SSL server certificate.</p> <p>Example: Server-Cert</p>

Table 1-5 Description of parameters defined in the PortalEnroll module *(Continued)*

Parameter	Description
<code>ldap.ldapauth.authtype</code>	<p>Specifies the authentication type—basic authentication or SSL client authentication—required to communicate with the portal directory.</p> <p>Permissible values: <code>BasicAuth</code> or <code>SslClientAuth</code>.</p> <ul style="list-style-type: none"> <code>BasicAuth</code> specifies basic authentication. If you choose this option, be sure to enter the correct values for <code>ldap.ldapauth.bindDN</code> and <code>password</code> parameters; the server uses the DN from the <code>ldap.ldapauth.bindDN</code> attribute to bind to the directory (default). <code>SslClientAuth</code> specifies SSL client authentication. If you choose this option, be sure to set the value of the <code>ldap.ldapconn.secureConn</code> parameter to <code>true</code> and the value of the <code>ldap.ldapauth.clientCertNickname</code> parameter to the nickname of the certificate to be used for SSL client authentication. <p>Example: <code>BasicAuth</code></p>
<code>ldap.basedn</code>	<p>Specifies the base DN for searching the portal directory—the server uses the value of the <code>uid</code> field from the HTTP input (what a user enters in the enrollment form) and the base DN to construct an LDAP search filter.</p> <p>Permissible values: Any valid DN string of up to 255 characters.</p> <p>Example: <code>O=siroe.com</code></p>
<code>ldap.objectclass</code>	<p>Specifies the object class to modify or update in the portal directory.</p> <p>Permissible values: Must be <code>inetOrgPerson</code> for the default portal enrollment form; see “Enrollment Forms” on page 57.</p> <p>Example: <code>inetOrgPerson</code></p>
<code>ldap.minConns</code>	<p>Specifies the minimum number of connections permitted to the directory.</p> <p>Permissible values: 1 to 3</p> <p>Example: 2</p>
<code>ldap.maxConns</code>	<p>Specifies the maximum number of connections permitted to the directory.</p> <p>Permissible values: 3 to 10</p> <p>Example: 10</p>

SSOAuthentication Plug-In Module

CMS provides a Single Sign-On (SSO) authentication module for user authentication. The Sun™ ONE Identity Server 6.0 will be integrated with the Certificate Server SSO authentication mechanism. This integration will make it possible for an Identity Server user to authenticate himself to the Certificate Server by providing his Single Sign-On token instead of userID and password. The user can also apply for a general-purpose user certificate with a single click of a button, eliminating the need to manually import or install the certificate. The user clicks the GetMyCert button in the Identity Server user profile page to automatically generate the user certificate.

The following section provides instructions for configuring the Certificate Server Single Sign-On (SSO) Authentication module to work with Identity Server 6.0.

NOTE	At the time of this writing, Identity Server 6.0 is not yet released. When it becomes available, please see the documentation that comes with that product for detailed information on configuring Identity Server to work with Certificate Server.
-------------	---

Configuring SSOAuthentication

Enabling this feature is a three-part procedure. Part 1 is described in detail in this document. For details on Parts 2 and 3, please see the Identity Server 6.0 documentation when it becomes available.

1. Create an instance of SSOBasedAuthentication in CMS.
2. In Identity Server, configure the Security service to work in non-SSL Certificate Server enrollment.
3. In Identity Server, configure the Security service to work in non-SSL Certificate Server enrollment.

Before You Begin

- Certificate Server 4.7 must be running and a Certificate Administrator must already be configured.
- Identity Server 6.0 must be installed and running.

Create an instance of SSOBasedAuthentication in CMS.

1. In the Certificate Server window, click Configuration>Authentication>Add.

2. In the Select Authentication Plug-in Implementation window, select `SSOBasedAuthentication`, and then click Next.
3. In the Authentication Instance Editor, provide the following information:
 - `com.ipplanet.am.naming.url`.** This is the Universal Resource Identifier (URI) for the Identity Server Naming Service. Type a URL to be used in place of the default URI. Use the following form:
`http://Identity_Server_root:portNumber/amserver/namingservice`
 - password.** Type the Shared Secret used by the Identity Server.
 - `com.ipplanet.am.cookie.name`.** Type the Cookie property used by Identity Server. The default is `iplanetDirectoryPro`.
 - `com.ipplanet.am.pcookie.name`.** Type the PCookie property used by Identity Server. The default is `DProPcookie`.
 - `com.ipplanet.am.services.deploymentDescriptor`.** Type the Deployment Descriptor property used by Identity Server. The default value is `amserver`.
4. Click OK.

Certificate-Based Enrollment

Certificate Management System supports certificate-based enrollment for browser certificates. End users can use preissued certificates to authenticate to the server in order to enroll for certificates. Below are two deployment scenarios that explain the usefulness of certificate-based enrollment.

- You have deployed a client that can generate dual key pairs and you want to issue dual certificates (one for *signing* and another for *encrypting* data) to your users. You also want to make sure that users put their key materials only on hardware tokens.

One way to achieve this would be to initialize hardware tokens in bulk and preload them with dual certificates issued by Certificate Management System for dual key pairs. You generate these certificates with some generic-looking common names, for example, `hardwaretoken1234`. This way, there's no one-to-one relation between users and the hardware tokens initially. Once the tokens are ready, you make them available to users by some means, for example, from a vending-machine-like box in the break room. Basically, a user can get and use any pre-initialized and certificate-loaded hardware token.

Next, each user uses the randomly-picked token to enroll (strictly speaking, renew) for a pair of certificates that have a subject name derived from their LDAP attribute values; the certificates will be issued for the existing key pairs preloaded into the token, but now the key pairs will be associated with the user's identity.

- You want users use the signing certificate already in their possession to get an encryption certificate.

For example, assume you have deployed Certificate Management System and have issued single certificates (for single key pairs) to users. Recently, you deployed a client application (such as Netscape Personal Security Manager) that is capable of generating dual key pairs. Your CMS installation includes the Data Recovery Manager, but you weren't using it until now because you didn't have clients that were capable of generating dual-key pairs. Now, you want your users to use their signing certificates as authentication tokens to request another certificate that they'll use for encrypting data.

To enable you to configure Certificate Management System for certificate-based enrollment, the following three enrollment forms are provided:

- `CertBasedDualEnroll.html`—this form enables end users to request dual certificates—one for signing another for encryption—by submitting pre-issued certificates as authentication tokens; when a user enrolls for a certificate, the server verifies the CA that has issued the certificate the user uses for authentication, uses the configured directory to formulate subject names for the new certificates, and issues the certificates.
- `CertBasedEncryptionEnroll.html`—this form is provided as a sample. It enables end users to request encryption certificates by submitting pre-issued certificates as authentication tokens; when a user enrolls for a certificate, the server verifies the CA that has issued the certificate the user uses for authentication, uses the configured directory to formulate the subject name for the new certificate, and issues the certificate.
- `CertBasedSingleEnroll.html`—this form is provided as a sample. It enables end users to request signing certificates by submitting pre-issued certificates as authentication tokens; when a user enrolls for a certificate, the server verifies the CA that has issued the certificate the user uses for authentication, uses the configured directory to formulate the subject name for the new certificate, and issues the certificate.

Note that all three enrollment forms by default work with the directory-based authentication module, named `UidPwdDirAuth`, explained in “`UidPwdDirAuth` Plug-in Module” on page 24. You can use the certificate-based enrollment forms with any of the authentication modules, for example, directory- and PIN-based or

NIS-server based authentication modules. However, this would require you to add the necessary hidden fields or variables to enrollment form that's provided for the corresponding authentication module; check Table 1-6 on page 59 to figure out which enrollment form works with which module.

In general, the following three hidden variables distinguish certificate-based enrollment forms from other enrollment forms:

- `certauthEnroll`—this variable specifies whether certificate-based enrollment is turned on or off.
- `certauthEnrollType`—this variable specifies one of the three certificate-based-enrollment types: `dual`, `single`, or `encryption`; `dual` specifies that the enrollment request is for dual certificates; `single` specifies that the enrollment request is for a signing certificate; and `encryption` specifies that the enrollment request is for an encryption certificate.

Note that choosing `dual` would require a client that's capable of generating dual key pairs.

- `doSslAuth`—this variable specifies whether the server should request the client for SSL client authentication. You must set the value of this parameter to `on` and make sure that the port number specified in the authentication instance is an SSL port.

Before modifying a form, be sure to take a look at the default certificate-based enrollment forms. Also check the customization-related information for the enrollment forms in *CMS Customization Guide*.

In addition to the enrollment forms, a policy plug-in named `IssuerConstraints` is also provided; see “IssuerConstraints Plug-in Module” on page 98. This plug-in allows you to configure the server to recognize the CA that issues the certificates that your users will use for authentication purposes; you need this policy to ensure that the CA issues certificates only to those users who present a valid certificate during enrollment. Note that in the current implementation, the CA that issues the new certificates must be the same as the one that issues the certificates users will use for authentication. That is, the issuer DN in the authentication certificate must match the issuer DN specified in the policy configuration.

Here are a few things to keep in mind:

- Enrollment requests for dual certificates must be submitted directly to the Certificate Manager; the Registration Manager doesn't support generation of dual certificates.

- The Certificate Manager provides a bulk-enrollment interface, which can be used to preload keys and certificates on hardware tokens before distributing them to users for certificate enrollment. For details, see section “Bulk Enrollment Interface” of *CMS Customization Guide*.
- When using certificate-based enrollment, the `IssuerConstraints` policy must be enabled and configured to check the CA (its issuer DN) in certificates users will use to authenticate to the server. Also, the value assigned to the `issuerDN` parameter must match the issuer DN of the CA that was used to generate hardware tokens in bulk.
- Enabling certificate-based enrollment creates one link, named `Certificate`, under the list of user-enrollment links in the end-entity enrollment interface. By default, the link points to the `CertBasedDualEnroll.html` form. If you want to use either of the other two forms, `CertBasedEncryptionEnroll.html` or `CertBasedSingleEnroll.html`, you should associate the `Certificate` link to the form you want to use or add more links to the `index.html` file.

General guidelines to set up certificate-based enrollment (for dual certificates) are as follows:

- On the server side you need do the following:
 - Customize the enrollment form you want your users to use for enrollment.
 - Enable the appropriate enrollment option, such as directory-based enrollment or NIS-server based enrollment. Be sure to configure the authentication module to compose the desired DN pattern.
 - Enable the Key Usage extension policy explained in “KeyUsageExt Plug-in Module” on page 189.

Take a look at the key-usage policy rule named `ClientCertKeyUsageExt` and see if it needs any modifications. For example, to get a signing-only certificate, you need to turn off `keyEncipherment` and `dataEncipherment` bits of the extension; similarly, to get an encryption-only certificate, you may need to turn off the `digitalSignature` bit of the extension.

- Configure the `IssuerRule` policy with the correct issuer DN and set the predicate expression so that the rule is applied to client certificates only.
- On the client side, you need to do the following:
 - Install drivers for the hardware tokens you want to use during bulk generation of key pairs and corresponding certificates with generic subject names.

- If you want to issue dual certificates, install a client that can generate dual key pairs; for example, Netscape Communicator (version 4.7 or later) with Netscape Personal Security Manager.

Enrollment Forms

The end-entity interface of the Certificate Manager and the Registration Manager include default HTML forms for all the authentication methods—manual and automated—supported by the server.

Enrollment forms can be categorized into two types, depending on the authentication method they support.

- Manual enrollment forms—these forms work with the built-in manual authentication module (see “Manual Authentication” on page 23), enabling users to request all types of certificates such as client certificates, server certificates, object-signing certificates, CA certificates, and so on. Manual enrollment for end users requires them to enter information such as name, email ID, department, organization, and the state and the country in which the organization is located, and submit the request for a personal certificate. Manual enrollment for server certificates requires the server administrator to paste the certificate signing request (in the PKCS#10 format) from the server into the specified area in the enrollment form; see Chapter 24, “Issuing and Managing Server Certificates” of *CMS Installation and Setup Guide*.

Because the Certificate Manager or Registration Manager cannot verify the information an end user or administrator enters against anything, it builds the certificate request based on the user input and puts the request in the agent queue for approval.

- Automated enrollment forms—these forms work with the corresponding plug-in module, enabling users to request certificates by authenticating to the configured repository, for example an LDAP directory or NIS directory.

All enrollment forms are accessible from the Enrollment tab of the End Entity Services interface. Note that by default the Enrollment tab lists only those forms that are associated with the manual enrollment method (it does not list the forms provided for the automated-enrollment methods). However, when you create an instance of any of the authentication modules provided for automated end-user enrollment—for example, directory-based or NIS-server based authentication—a link to the corresponding form is automatically created under the Browser section of the Enrollment tab. Instructions for enabling automated end-user enrollments is covered in Chapter 15, “Setting Up End-User Authentication” of *CMS Installation and Setup Guide*.

Before asking users to use any of the enrollment forms, you should review the form and make the appropriate changes to the form content. For example, some of the forms include instructions for administrators and you should delete those lines. Files for all enrollment forms are located here:

```
<server_root>/cert-<instance_id>/web/ee
```

For basic instructions to customize any of the enrollment forms, see section “Step 5. Set Up the Enrollment Interface” in Chapter 15, “Setting Up End-User Authentication” of *CMS Installation and Setup Guide*.

For advanced information on customizing the HTML forms, including how to make changes to the embedded Javascript functions, see *CMS Customization Guide*.

Figure 1-10 shows the End Entity Services interface of a Certificate Manager with the manual enrollment form for end users selected.

Figure 1-10 End Entity Services interface of a Certificate Manager

The screenshot displays the iPlanet Certificate Management System interface. The top header bar is purple with the text "iPlanet® Certificate Management System" on the left and "Certificate Manager" on the right. Below the header, there are four tabs: "Enrollment", "Renewal", "Revocation", and "Retrieval". The "Enrollment" tab is selected. On the left side, there is a vertical navigation menu with the following items: "Browser", "Manual" (highlighted in blue), "Server", "SSL Server", "Registration Manager", "Certificate Manager", "OCSP Responder", "Other", "Object Signing (Browser)", "Object Signing (PKCS10)", "CMC", and "Enrollment". The main content area is titled "Manual User Enrollment" and contains the following text: "Use this form to submit a request for a personal certificate. After you click the Submit button, your request will be submitted to an issuing agent for approval. When an issuing agent has approved your request you will receive the certificate in email, along with instructions for installing it." Below this text is an "Important:" note: "Be sure to request your certificate on the same computer on which you plan to use the certificate." Further down is a section titled "User's Identity" with the instruction: "Enter values for the fields you want to have in your certificate. Your site may require you to fill in certain fields. (* = required field)". This section contains five text input fields: "* Full name:", "Login name:", "Email address:", "Organization unit:", and "Organization:". At the bottom, there is a "Country:" label followed by a dropdown menu.

Table 1-6 lists the forms that correspond to the menu options in the Enrollment tab of the End Entity Services interface of the Certificate Manager and Registration Manager.

Table 1-6 Default forms for end-entity enrollment

Menu link and form filename	Description
Browser (This section lists menu options for end-user enrollments.)	
Manual (ManUserEnroll.html)	End users can use this form to request SSL client and S/MIME certificates. Requests submitted using this form get queued for agent approval.
Directory (DirUserEnroll.html)	This form works with the <code>UidPwdDirAuth</code> module, enabling end users to request SSL client and S/MIME certificates by entering their user IDs and passwords for the directory; the server verifies this information against the configured directory and issues the certificate.
Directory and PIN (DirPinUserEnroll.html)	This form works with the <code>UidPwdPinDirAuth</code> module, enabling end users to request SSL client and S/MIME certificates by entering their user IDs, passwords, and PINs for the configured directory; the server verifies this information against the specified directory and issues the certificate.
NIS (NISUserEnroll.html)	This form works with the <code>NISAuth</code> module, enabling end users to request SSL client and S/MIME certificates by entering their NIS user IDs and passwords for the configured NIS server.
Portal (PortalEnrollment.html)	<p>This form works with the <code>PortalEnroll</code> module, enabling end users to register for an online service and at the same time submit a request for a personal certificate. Note that the form models the standard LDAP object class <code>inetOrgPerson</code>, which has many useful attributes that can be used in a real portal deployment.</p> <p>As a part of registration, a user is required (by the portal authentication module) to supply a user ID and password for user ID validation and a first and last name for user registration. Entering information in other fields are optional; the server retrieves the rest of the information needed to construct the subject name for the certificate from the directory. As explained in “PortalEnroll Plug-in Module” on page 44, if the user ID is unique, the server issues a certificate and registers the user automatically. To protect the privacy of a user’s password, the server turns it in to a SHA-1 or MD5 hashed password before storing it in the directory.</p>

Table 1-6 Default forms for end-entity enrollment (*Continued*)

Menu link and form filename	Description
Certificate (CertBasedDualEnroll.html)	<p>This form by default works with the <code>UidPwdDirAuth</code> module, enabling end users to request dual certificates (one for signing another for encryption) by submitting pre-issued certificates as authentication tokens; the server verifies the CA that has issued the certificate, uses the configured directory to formulate the subject names for the new certificates, and issues the certificate.</p> <p>Note that the link appears only if you create an instance of the <code>UidPwdDirAuth</code> module and if the port number specified in the instance configuration is an SSL port. For details, see “Certificate-Based Enrollment” on page 53.</p>
Server (This section lists menu options for SSL server, Registration Manager, Certificate Manager, and OCSP Responder enrollments.)	
SSL Server (ManServerEnroll.html)	Server administrators can use this form to request SSL server certificates for SSL-enabled servers, such as iPlanet Administration Server and Directory Server. Requests submitted using this form get queued for agent approval.
Registration Manager (ManRAEnroll.html)	Registration Manager administrators can use this form to request a signing certificate for a Registration Manager; see section “Signing Key Pair and Certificate” in Chapter 14, “Managing CMS Keys and Certificates” of <i>CMS Installation and Setup Guide</i> . Requests submitted using this form get queued for agent approval.
Certificate Manager (ManCAEnroll.html)	<p>Certificate Manager administrators can use this form to request <i>CA signing certificates</i> for Certificate Managers functioning as subordinate CAs; see section “CA Signing Key Pair and Certificate” in Chapter 14, “Managing CMS Keys and Certificates” of <i>CMS Installation and Setup Guide</i>. Requests submitted using this form get queued for agent approval.</p> <p>Only the Certificate Manager provides this form.</p>
OCSP Responder (OCSPResponder.html)	Server administrators can use this form to enroll for an OCSP responder certificate. Requests submitted using this form get queued for agent approval.
WTLS (This section lists menu options for wireless certificate enrollments, and the section appears only if you configured the Certificate Manager to support issuance of Wireless Transport Layer Support (wTLS)-compliant certificates during installation.)	
Client (WTLSTManUserEnroll.html)	<p>Users can use this form to enroll for a wireless certificate.</p> <p>Only the Certificate Manager provides this form.</p>

Table 1-6 Default forms for end-entity enrollment (*Continued*)

Menu link and form filename	Description
Server (WTLSManServerEnroll.html)	Server administrators can use this form to enroll for a wireless certificate; the certificate request can be in PKCS#10 format. Only the Certificate Manager provides this form.
Other (This section lists menu options for object signing enrollments.)	
ObjectSigning (PKCS10) (ObjSignPKCS10Enroll.html)	Server administrators can use this form to enroll for a certificate (by submitting the request in the PKCS #10 format) that allows them to sign objects, such as Java applets. Requests submitted using this form get queued for agent approval.
ObjectSigning (Browser) (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. Requests submitted using this form get queued for agent approval. Note that when issuing an object signing certificate to Microsoft IE, if you need to generate a .CER and a .PVK file for use by Microsoft signcode tool, follow the instructions in “Generating Files Required By Third-Party Object Signing Tools” on page 63.
CMC (CMCResponder.html)	End users and administrators can use this form to submit a certificate request in the CMC format.

Customizing Enrollment Forms for Generating DSA Key Pairs

Netscape Communicator (version 4.x and later) can successfully obtain and use DSA client certificates for SSL client authentication. These versions of Communicator can also recognize the signature on SSL certificates signed by a DSA CA. In order for Communicator to generate a DSA key pair, you must modify the `KEYGEN` tag in the default certificate enrollment forms; the modifications will indicate that the DSA algorithm is to be used, and will also supply the PQG parameters. For details on the `KEYGEN` tag, see the document entitled *Netscape Extensions for User Key Generation* available at this site:

<http://home.netscape.com/eng/security/comm4-keygen.html>

Depending on the enrollment plug-in you want to use for authenticating end users, you may need to modify the `KEYGEN` tags in the following certificate enrollment forms:

- DirPinUserEnroll.html

- DirUserEnroll.html
- ManObjSignEnroll.html
- ManUserEnroll.html
- NISUserEnroll.html
- PortalEnrollment.html

These files are located in this directory:

```
<server_root>/cert-<instance_id>/web/ee
```

The procedure below explains how to modify an enrollment form to generate a DSA key pair when used with Netscape Communicator:

1. Go to the configuration directory of the Certificate Manager:
`<server_root>/cert-<instance_id>/config`
2. Open the Certificate Manager's configuration file (`CMS.cfg`) in a text editor.
3. Open the enrollment form in a text editor.
4. In the configuration file, find the `DSSParms` entry; this entry represents the PQG attribute and its value contains the PQG parameters that the CA has generated during configuration.
5. Copy the value of the `DSSParms` entry.
6. Go to the text editor that has the enrollment form open.
7. Search for the `KEYGEN` tag.
8. Insert the cursor at the end of the word `KEYGEN`, add a space after the word `KEYGEN`, and type the following:

```
keytype="DSA" PQG=
```

9. Paste the value of the `DSSParms` entry following the equal to (=) sign and enclose the value you pasted in double quotes (" ").

An example of a modified `KEYGEN` tag is shown below (the modifications are shown in bold):

```
<KEYGEN keytype="DSA"
PQG="MIIBHgKBgQCsQeVqw5ID/xhSe7s4vLaOuKskCFJN23OBgWCEquYIZbMZdHN
7015p6nN7XsDpTWBccLdrSdpMxmJd8rF2agb3tbk9hjZ6//MfLCTAwvegdgAzzRw
B7akOgYD/SpPFb7rYuvPfkirjiDDrrp9r+csWqnue9uABvJtWGNW8WVYP6wIVAMC
Ru0u3q+PORrJxO9QcswzrLpnfAoGAM3ZBjxLTPbXOGWIXHZnIFSpGAWlJzK5ywEt
nabJWfiIRrWi3hyWLj98PcIc2cxbpOh60rwqeELUMv74V72Q2+HwIQwsPvTFyQUc
BtOG40z1XoFwEqLaqDoXv3iA0Zp2XQy/JQFbx23J+0HKz7iB7co04Lca0wDU7Z0x
+oTwmsd0=" name="subjectKeyGenInfo">
```

10. Repeat steps 7 through 9 to modify any additional `KEYGEN` tags.
11. Save your changes.
12. Next, configure the Certificate Manager to accept DSA key based certificate enrollment requests.

A Certificate Manager by default only accepts RSA key-based requests. For the server to accept DSA key based certificate requests, the value of the `algorithms` parameter in the `KeyAlgRule` policy rule must be set to `RSA, DSA`. For instructions to change policy rules, see Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*.

Generating Files Required By Third-Party Object Signing Tools

When issuing an object-signing certificate to Microsoft IE, Certificate Management System can generate a certificate (`.CER`) and a private key (`.PVK`) files for use by Microsoft `signcode` tool or any third-party sign tools that rely on these files. For the server to generate these files, you must edit the default form provided for requesting an object-signing certificate for browsers.

To generate the private-key file:

1. Go to this directory: `<server_root>/cert-<instance_id>/web/ee`
2. Locate the file named `ManObjSignEnroll.html`.
3. Open it in an editor.
4. Search for this line:

```
Enroll.GenKeyFlags = 1          ' key exportable
```

5. Type the following line below it:

```
Enroll.PVKFilename = "<pvk_file_path>"
```

Your changes should look like this:

```
...
Enroll.GenKeyFlags = 1          ' key exportable
Enroll.PVKFilename = "<pvk_file_path>"
szCertReq = Enroll.createPKCS10(szName, "1.3.6.1.5.5.7.3.2")
...
```

6. Replace `<pvk_file_path>` with the absolute path, including the filename, to the directory in which you want the private key file created; for example, `"C:\myKey.PVK"`. Be sure to use the `.PVK` extension and to enclose the path in double quotes.
7. Optionally, you may further edit the form to include a text field for entering the file path.
8. Save your changes.
9. Now use the form to issue an object-signing certificate.

If your users need to generate Software Publishing File (SPC) files for their object-signing certificates, you should ask them to use the Microsoft tool named `cert2spc`. The SPC file enables them to execute commands such as this:

```
signcode -spc myCert.spc -v myKey.pvk file.exe
```

Here's how a user can create a SPC file for an object-signing certificate:

1. Open a web browser window.
2. Go to the End Entity Services interface.
3. Locate the object-signing certificate for which you want to create the SPC file.
4. Scroll down the page (that shows the certificate information in detail) to find the certificate in base-64 encoded format. It looks like this:

```
-----BEGIN CERTIFICATE-----
```

```
MIICJzCCAZCgAwIBAgIBAzANBgkqhkiG9w0BAQQFADBCCSAwHgYDVQQKEXdOZXRz
Y2FwZSBDb21tdW5pY2F0aW9ucznghnMVQ2VydGhmaWNhdGUgQXV0aG9yaXR5MB4
XDtk4MDgyNzE5MDwMFOXDtk5MDIyMzE5MDAwMnBjdGngYoxIDAeBgNVBAoTF05ld
HNjYXBleENvbnV1bm1jYXRpb25zMQ8wDQYDVQQLEWZQZW9wbGUxZzAVBgoJkiaJk
IsZAEBEwdzdXByaXlhMRcwFQYDVQQDw5TdXByaXlhIFNoZXR0eTEjMCEGCSqGSIb
3DondgJA
```

```
-----END CERTIFICATE-----
```

5. Create an ASCII file named `cert.b64`.
6. Copy and paste the base-64 encoded certificate blob, including the marker lines `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` to the file.
7. Convert the text-based certificate to its DER-encoded format using the ASCII to Binary tool, explained in *CMS Customization Guide*.

For example, the command

```
<server_root>/bin/cert/tools/AtoB cert.b64 cert.der
```


converts the base-64 encoded certificate in the `cert.b64` file to its DER-encoded format and writes the DER-encoded certificate to a file named `cert.der`.

8. Next, use the Microsoft tool named `cert2spc` to convert the DER-encoded certificate to SPC format. For example, the command

```
cert2spc cert.der cert.spc
```

converts the DER-encoded certificate in the `cert.der` file to its SPC format and writes the certificate to a file named `cert.spc`.

For additional information, check these links:

- <http://www.lantimes.com/ltparts/connect/shoptalk1.htm>
- <http://www.thawte.com/certs/developer/msauthenticode.html>
- <http://www.drh-consultancy.demon.co.uk/pkcs12faq.html>

Job Plug-in Modules

iPlanet Certificate Management Server (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.

This chapter describes the job plug-in modules that are provided with Certificate Management System and explains how to schedule times for jobs.

The chapter has the following sections:

- Overview of Job Plug-in Modules (page 67)
- RenewalNotificationJob Plug-in Module (page 69)
- RequestInQJob Plug-in Module (page 73)
- UnpublishExpiredJob Plug-in Module (page 76)
- Customizing Notification Messages (page 81)

Overview of Job Plug-in Modules

Both the Certificate Manager and Registration Manager provide a set of job plug-in modules that can be employed by the server to automate certain activities. The Job Plug-in Registration tab of the CMS window (see Figure 2-1) lists all the modules and the corresponding classes that are currently registered with a Certificate Manager.

Figure 2-1 Default job modules for the Certificate Manager

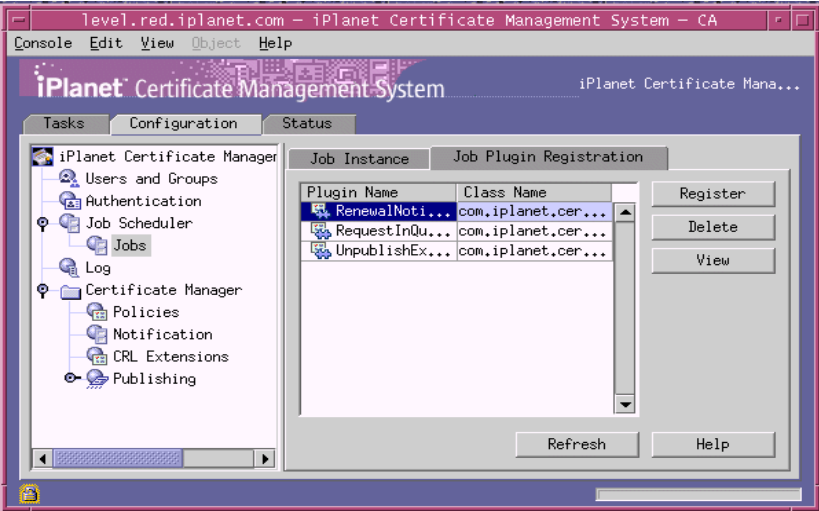


Table 2-1 lists these modules.

Table 2-1 Schedulable job plug-in modules for Certificate Manager and Registration Manager

Plug-in module name	Description
RenewalNotificationJob	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 “RenewalNotificationJob Plug-in Module” on page 69.
RequestInQueueJob	A schedulable job that notifies agents at regular intervals of the current state of the request queue. 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 “RequestInQJob Plug-in Module” on page 73.
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 “UnpublishExpiredJob Plug-in Module” on page 76.

Jobs are implemented as Java classes, which are then registered with Certificate Management System as plug-in modules. You can use a given implementation of a job 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.

Note that the name of the Java class for a job plug-in module is in this format:

```
com.iplanet.certsrv.jobs.<plugin_name>
```

where <plugin_name> is the name of a plug-in module. For example, the Java class for the `RenewalNotificationJob` module would be:

```
com.iplanet.certsrv.jobs.RenewalNotificationJob
```

RenewalNotificationJob Plug-in Module

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.

The `RenewalNotificationJob` plug-in module 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>/cert-<instance_id>/emails`

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 81.

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 80.
- 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.

Configuration Parameters of `RenewalNotificationJob`

In the CMS configuration file, the `RenewalNotificationJob` module is identified as `jobsScheduler.impl.RenewalNotificationJob.class=com.iplanet.certsrv.jobs.RenewalNotificationJob`.

In the CMS window, the module is identified as `RenewalNotificationJob`. Figure 2-2 shows how the configurable parameters pertaining to the plug-in module are displayed in the CMS window.

Figure 2-2 Parameters defined in the RenewalNotificationJob module

Job Instance Editor

Job Instance ID: certRenewalNotifier
Job Plugin ID: RenewalNotificationJob

enabled ☒

cron 0 3 * * 1-5

notifyTriggerOffset 30

notifyEndOffset 30

senderEmail CertCentral@siroe.com

emailSubject Certificate Renewal Notification

emailTemplate d:/Netscape/Server4/cert-testCA/emails/rnJob1.txt

summary.enabled ☒

summary.recipientEmail ca_agent1@siroe.com,ca_agent2@siroe.com

summary.senderEmail CAAdmin@siroe.com

summary.emailSubject Certificate Renewal Notification Summary

summary.itemTemplate d:/Netscape/Server4/cert-testCA/emails/rnJob1Item.txt

summary.emailTemplate d:/Netscape/Server4/cert-testCA/emails/rnJob1Summary.txt

Sender email address of summary

OK Cancel Help

Table 2-2 gives details about each of these parameters.

Table 2-2 Description of parameters defined in the RenewalNotificationJob module

Parameter	Description
enabled	Specifies whether the job is enabled or disabled. Check the box to enable the job. Uncheck the box to disable the job. If you enable the job and set the remaining parameters correctly, the server runs the job at scheduled intervals.
cron	<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>Permissible values: Must follow the convention specified in “Schedule for Executing Jobs” on page 80.</p> <p>Example: 03 * * 1-5</p>

Table 2-2 Description of parameters defined in the RenewalNotificationJob module *(Continued)*

Parameter	Description
<code>notifyTriggerOffset</code>	<p>Specifies how long (in days) before certificate expiration the first notification will be sent.</p> <p>Permissible values: As applicable.</p> <p>Example: 30</p>
<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>Permissible values: As applicable.</p> <p>Example: 30</p>
<code>senderEmail</code>	<p>Specifies the sender of the notification messages (who will be notified of any delivery problems).</p> <p>Permissible values: The complete email address.</p> <p>Example: CertCentral@siroe.com</p>
<code>emailSubject</code>	<p>Specifies the subject line of the notification message.</p> <p>Permissible values: An alphanumeric string of up to 255 characters.</p> <p>Example: Certificate Renewal Notification</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>Permissible values: Template file path, including the file name.</p> <p>Example: C:\iplanet\servers\cert-testCA\emails\renewJob.txt</p>
<code>summary.enabled</code>	<p>Specifies whether a summary report of renewal notifications should be compiled and sent. Check the box if you want the server to compose and send a summary report. Uncheck the box if you don't want the server to compose and send a summary report. If you check the box, be sure to set the remaining parameters; these are required by the server to send the summary report.</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>Permissible values: Full email addresses, separated by commas.</p> <p>Example: ca_agent1@siroe.com,ca_agent2@siroe.com</p>
<code>summary.senderEmail</code>	<p>Specifies the sender of the summary message (who will be notified of any delivery problems).</p> <p>Permissible values: The full email address.</p> <p>Example: CAAdmin@siroe.com</p>

Table 2-2 Description of parameters defined in the RenewalNotificationJob module *(Continued)*

Parameter	Description
summary. emailSubject	Specifies the subject line of the summary message. Permissible values: An alphanumeric string of up to 255 characters. Example: Certificate Renewal Notification Summary
summary. itemTemplate	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 summary.emailTemplate parameter below). For details, see “Customizing Notification Messages” on page 81. Permissible values: The template file path, including the file name. Example: C:\iplanet\servers\cert-testCA\emails\renewJobItem.txt
summary. emailTemplate	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 81. Permissible values: The template file path, including the file name. Example: C:\iplanet\servers\cert-testCA\emails\renewJobSummary.txt

RequestInQJob Plug-in Module

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.

The RequestInQJob plug-in module 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>/cert-<instance_id>/emails
```

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 80.
- 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.

Configuration Parameters of RequestInQJob

In the CMS configuration file, the `RequestInQJob` module is identified as `jobsScheduler.impl.RequestInQJob.class=com.iplanet.certsrv.jobs.RequestInQJob`.

In the CMS window, the module is identified as `RequestInQJob`. Figure 2-3 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 2-3 Parameters defined in the RequestInQJob module

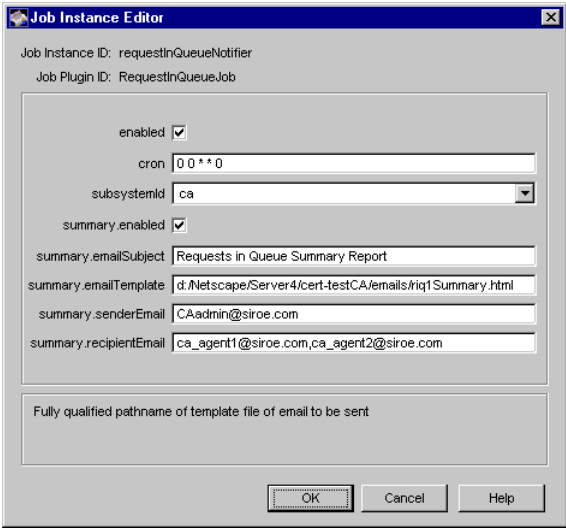


Table 2-3 gives details for each of these parameters.

Table 2-3 Description of parameters defined in the RequestInQJob module

Parameter	Description
enabled	Specifies whether the job is enabled or disabled. Check the box to enable the job. Uncheck the box to disable the job. If you enable the job and set the remaining parameters correctly, the server runs the job at scheduled intervals.
cron	<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>Permissible values: Must follow the convention specified in “Schedule for Executing Jobs” on page 80.</p> <p>Example: <code>00**0</code></p>
subsystemid	<p>Specifies the subsystem that this job is for.</p> <p>Permissible values: <code>ca</code> or <code>ra</code>.</p> <ul style="list-style-type: none">• <code>ca</code> specifies that the job is for the Certificate Manager.• <code>ra</code> specifies that the job is for the Registration Manager. <p>Example: <code>ca</code></p>

Table 2-3 Description of parameters defined in the RequestInQJob module *(Continued)*

Parameter	Description
summary.enabled	Specifies whether a summary of the job accomplished should be compiled and sent. Check the box if you want the server to compose and send a summary report. Uncheck the box if you don't want the server to compose and send a summary report. If you check the box, be sure to set the remaining parameters; these are required by the server to send the summary report.
summary.emailSubject	Specifies the subject line of the summary message. Permissible values: An alphanumeric string of up to 255 characters. Example: Summary Report of Requests in the Agent Queue
summary.emailTemplate	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 81. Permissible values: The template file path, including the file name. Example: C:\iplanet\servers\cert-testCA\emails\reqInQJobSummary.txt
summary.senderEmail	Specifies the sender of the notification message (who should be notified of any delivery problems). Permissible values: The full email address. Example: CAadmin@siroe.com
summary.recipientEmail	Specifies the recipients of the summary message. These should be, for example, agents who need to process pending requests. Permissible values: Full email addresses, separated by commas. Example: ca_agent1@siroe.com,ca_agent2@siroe.com

UnpublishExpiredJob Plug-in Module

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.

The `UnpublishExpiredJob` plug-in module 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>/cert-<instance_id>/emails
```

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 83.

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 80.
- 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 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.

Note that the job automates removal of expired certificates from the directory. You can also remove expired certificates manually following the instructions outlined in section “Manually Updating Certificates and CRLs in a Directory” in Chapter 19, “Setting Up LDAP Publishing” of *CMS Installation and Setup Guide*.

Configuration Parameters of UnpublishExpiredJob

In the CMS configuration file, the UnpublishExpiredJob module is identified as `jobsScheduler.impl.ExpiredUnpublishJob.class=com.iplanet.certsrv.jobs.UnpublishExpiredJob`.

In the CMS window, the module is identified as UnpublishExpiredJob. Figure 2-4 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 2-4 Parameters defined in the UnpublishExpiredJob module

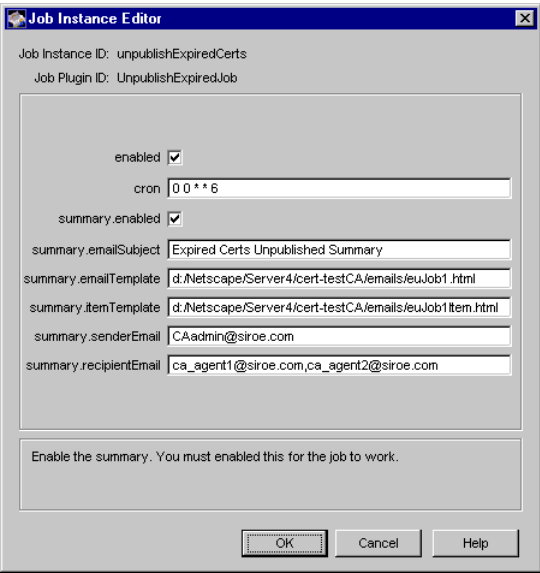


Table 2-4 gives details for each of these parameters.

Table 2-4 Description of parameters defined in the UnpublishExpiredJob module

Parameter	Description
enabled	Specifies whether the job is enabled or disabled. Check the box to enable the job. Uncheck the box to disable the job. If you enable the job and set the remaining parameters correctly, the server runs the job at scheduled intervals.
cron	<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>Permissible values: Must follow the convention specified in “Schedule for Executing Jobs” on page 80.</p> <p>Example: <code>00**6</code></p>
summary.enabled	Specifies whether a summary of the certificates removed by the job should be compiled and sent. Check the box if you want the server to compose and send a summary report. Uncheck the box if you don’t want the server to compose and send a summary report. If you check the box, be sure to set the remaining parameters; these are required by the server to send the summary report.
summary.emailSubject	<p>Specifies the subject line of the summary message.</p> <p>Permissible values: An alphanumeric string of up to 255 characters.</p> <p>Example: Expired Certificate Removal Job Summary</p>
summary.emailTemplate	<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 81.</p> <p>Permissible values: The template file path, including the file name.</p> <p>Example: <code>C:\iplanet\servers\cert-testCA\emails\unpublishCertsJobSummary.html</code></p>
summary.itemTemplate	<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>Permissible values: The template file path, including the file name.</p> <p>Example: <code>C:\iplanet\servers\cert-testCA\emails\unpublishCertsJobItem.txt</code></p>
summary.senderEmail	<p>Specifies the sender of the summary message (who should be notified of any delivery problems).</p> <p>Permissible values: The full email address.</p> <p>Example: <code>CAadmin@siroe.com</code></p>

Table 2-4 Description of parameters defined in the UnpublishExpiredJob module *(Continued)*

Parameter	Description
summary. recipientEmail	Specifies the recipients of the summary message. These can be, for example, agents who need to know the status of user certificates. Permissible values: Complete email addresses, separated by commas. Example: cert_agent1@siroe.com, cert_agent2@siroe.com

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 2-5, 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 2-5 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
```

Customizing Notification Messages

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>/cert-<instance_id>/emails
```

Both text and HTML templates are included by default. They are listed in Table 2-6.

Templates for Summary Notifications

Table 2-6 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 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 83.

Table 2-6 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 ExpiredUnpublishJob template.
Templates for RequestInQueueJob module	
riqlItem.html	Template for formatting the items to be included in the summary table, which is constructed using the riqlSummary.html template.
riqlSummary.html	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 RenewalNotificationJob module	
rnJob1.txt	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.
rnJob1Item.txt	Template for formatting the items to be included in the summary report, which is constructed using the rnJob1Summary.txt template.
rnJob1Summary.txt	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.

<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 83.

For example, a certificate-issuance-notification message can make use of tokens as follows:

```
-----
CERTIFICATE RENEWAL NOTIFICATION
-----

Your certificate will expire soon:

Serial Number= $SerialNumber
SubjectDN= $SubjectDN
IssuerDN= $IssuerDN
Validity Period= $NotBefore - $NotAfter

To renew your certificate, please follow this URL:

https://$HttpHost:$HttpPort

If you have any questions or problems, please send an email to
cert_central@siroe.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 Renewal Notification Messages
- Tokens for Request In Queue Notification Messages
- Tokens for Directory Update Notification Messages

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 2-7 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 2-8.

Table 2-7 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.
\$SummaryTotalNum	Specifies the total number of items (certificates that require to be renewed) in the summary report.
\$SummaryTotalSuccess	Specifies how many of the total number of items in the summary report succeeded.

Table 2-8 lists the tokens for the inner list items.

Table 2-8 Tokens for items in renewal-notification job's summary report

Token	Description
\$CertType	Specifies the type of certificate—whether SSL client (<i>client</i>), SSL server (<i>server</i>), Registration Manager's signing certificate (<i>ra</i>), Certificate Manager's CA signing certificate (<i>ca</i>), router certificate (<i>Cisco-router</i>), or other (<i>other</i>).
\$HttpHost	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 83.)
\$HttpPort	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 83.)
\$IssuerDN	Specifies the distinguished name of the certificate issuer.
\$NotAfter	Specifies the <i>NotAfter</i> attribute.
\$NotBefore	Specifies the <i>NotBefore</i> attribute.

Table 2-8 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; the serial number will be displayed as a hexadecimal value in the resulting message.
\$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 2-9 lists the tokens that you can use for formulating the content of the `RequestInQueueJob` job's summary report.

Table 2-9 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"> If the notification is sent by a Certificate Manager, this will be <code>ca</code>. If the notification is sent by a Registration Manager, this will be <code>ra</code>.
\$ExecutionTime	Specifies the time the job (instance) was run.
\$RecipientEmail	Specifies the email address of the recipient.
\$SenderEmail	Specifies the email address of the sender.
\$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 2-10 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 2-11.

Table 2-10 Tokens for the unpublish-expired job's summary report

Token	Description
\$InstanceID	Specifies the name of the job instance that generated this summary report.
\$ExecutionTime	Specifies the time the job (instance) was run.
\$SummaryItemList	Specifies the list of items in the summary notification. Each item corresponds to a certificate the job detects for removal from the publishing directory.
\$SummaryTotalFailure	Specifies the total number of items in the summary report that failed.
\$SummaryTotalNum	Specifies the total number of items (certificates to be removed from the directory) in the summary report.
\$SummaryTotalSuccess	Specifies how many of the total number of items in the summary report succeeded.

Table 2-11 lists the tokens for the inner list items.

Table 2-11 Tokens for items in the unpublish-expired job's summary report

Token	Description
\$CertType	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>).
\$IssuerDN	Specifies the distinguished name of the certificate issuer.
\$NotAfter	Specifies the <code>NotAfter</code> attribute.
\$NotBefore	Specifies the <code>NotBefore</code> attribute.
\$RequestorEmail	Specifies the requestor's email address.
\$SerialNumber	Specifies the serial number of the certificate; the serial number will be displayed as a hexadecimal value in the resulting message.
\$Status	Specifies whether the operation failed or succeeded.
\$SubjectDN	Specifies the distinguished name of the certificate subject.

Constraints Policy Plug-in Modules

You can configure iPlanet Certificate Management Server (CMS) to apply certain organizational policies to an end entity's certificate enrollment, renewal, and revocation requests before servicing them. For example, some of the policies you might want Certificate Management System to apply to 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.

Certificate Management System comes with various policy plug-in modules that define the formulation of a certificate's content and govern the server's certificate generation and management operations. The modules are categorized, based on their functionality, into two groups: constraints-specific policy modules and extension-specific policy modules.

This chapter explains the constraints-specific policy plug-in modules in detail—it lists and briefly describes the modules that are installed with Certificate Management System, and then explains each one in detail. For details on extension-specific modules, see Chapter 4, “Certificate Extension Plug-in Modules”.

The chapter has the following sections:

- Overview of Constraints-Specific Policy Modules (page 88)
- AttributePresentConstraints Plug-in Module (page 90)
- DSAKeyConstraints Plug-in Module (page 95)
- IssuerConstraints Plug-in Module (page 98)
- KeyAlgorithmConstraints Plug-in Module (page 101)
- RenewalConstraints Plug-in Module (page 103)
- RenewalValidityConstraints Plug-in Module (page 106)

- RevocationConstraints Plug-in Module (page 110)
- RSAKeyConstraints Plug-in Module (page 112)
- SigningAlgorithmConstraints Plug-in Module (page 115)
- SubCANameConstraints Plug-in Module (page 118)
- UniqueSubjectNameConstraints Plug-in Module (page 121)
- ValidityConstraints Plug-in Module (page 124)

Overview of Constraints-Specific Policy Modules

Constraints-specific policy plug-in modules help you define rules or constraints that Certificate Management System uses to evaluate an incoming certificate enrollment, renewal, or revocation request. Each module enables you to configure the server to check the request for particular attributes, and, based on the configured criteria, either modify these attributes or reject the request altogether.

Policy plug-in modules are implemented as Java classes and are registered in the CMS policy framework. The Policy Plugin Registration tab of the CMS window (Figure 3-1) lists all the modules that are registered with a CMS instance.

Figure 3-1 CMS window showing policy modules registered with a Certificate Manager

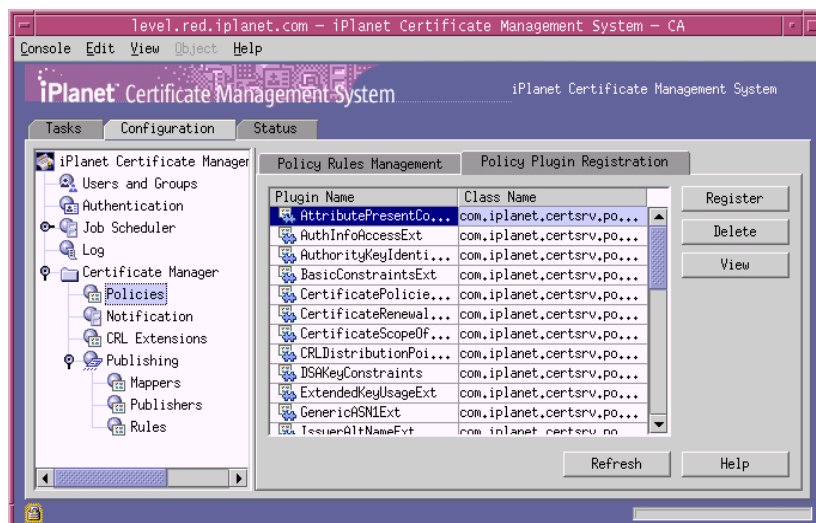


Table 3-1 lists constraints-specific policy modules that are installed with a Certificate Manager. An installation of a Registration Manager also includes all these modules, except for the ones noted below:

- IssuerConstraints
- SubCANameConstraints
- UniqueSubjectNameConstraints

Note that the name of the Java class for a policy plug-in module is in this format:

```
com.iplanet.certsrv.policy.<plugin_name>
```

where <plugin_name> is the name of a plug-in module. For example, the Java class for the `AttributePresentConstraints` module would be:

```
com.iplanet.certsrv.policy.AttributePresentConstraints
```

You can use whichever modules you need in order to define policy rules for a Certificate Manager or Registration Manager. Note that no modules are provided for the Data Recovery Manager. Both Certificate Manager and Registration Manager subject a request to policy checking as explained in section “Policy Processor” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*.

Keep in mind that the changes made to a request by a Registration Manager may be overwritten by a Certificate Manager when it subjects the request to its own policy checks.

Table 3-1 Default constraints-specific policy plug-in modules

Plug-in module name	Function
<code>AttributePresentConstraints</code>	Rejects a request if an LDAP attribute is not present in the enrolling user’s directory entry or if the attribute does not have a specified value. For details, see “AttributePresentConstraints Plug-in Module” on page 90.
<code>DSAKeyConstraints</code>	Certifies only those DSA keys that have specific key lengths. For details, see “DSAKeyConstraints Plug-in Module” on page 95.
<code>IssuerConstraints</code>	Checks for certificates that have been issued by a particular CA. For details, see “IssuerConstraints Plug-in Module” on page 98.
<code>KeyAlgorithmConstraints</code>	Certifies only those keys that are generated using one of the permitted algorithms, such as RSA or DSA. For details, see “KeyAlgorithmConstraints Plug-in Module” on page 101.
<code>RenewalConstraints</code>	Allows or rejects requests for renewal of expired certificates. For details, see “RenewalConstraints Plug-in Module” on page 103.

Table 3-1 Default constraints-specific policy plug-in modules *(Continued)*

Plug-in module name	Function
RenewalValidityConstraints	Enforces the number of days before which a currently active certificate can be renewed and sets a new validity period for the renewed certificate. For details, see “RenewalValidityConstraints Plug-in Module” on page 106.
RevocationConstraints	Allows or rejects requests for revocation of expired certificates. For details, see “RevocationConstraints Plug-in Module” on page 110.
RSAKeyConstraints	Certifies only those RSA keys that have specific key lengths. For details, see “RSAKeyConstraints Plug-in Module” on page 112.
SigningAlgorithmConstraints	Specifies the signature algorithm to be used by the CA (a Certificate Manager) to sign certificates. For details, see “SigningAlgorithmConstraints Plug-in Module” on page 115.
SubCANameConstraints	Checks for issuer name uniqueness and prevents a CA from issuing a subordinate CA certificate with issuer name same as its own. For details, see “SubCANameConstraints Plug-in Module” on page 118.
UniqueSubjectNameConstraints	Checks for certificate subject name uniqueness and prevents issuance of multiple certificates with same subject names. For details, see “UniqueSubjectNameConstraints Plug-in Module” on page 121.
ValidityConstraints	Checks whether the validity period of a certificate falls within a specific validity period. For details, see “ValidityConstraints Plug-in Module” on page 124.

AttributePresentConstraints Plug-in Module

The `AttributePresentConstraints` plug-in module implements the attribute present constraints policy. The module enables you to configure the Certificate Manager and Registration Manager to reject a request if an LDAP attribute (for example, `pin`) is not present in the enrolling user’s directory entry or if the attribute does not have a specified value. An example usage is in “Step 3. Enable the AttributePresentConstraints Policy” in Chapter 15, “Setting Up End-User Authentication” of *CMS Installation and Setup Guide*.

Note that many of the parameters defined in the module (see Table 3-2 on page 92) are specified in the same way as the modules provided for authenticating users during directory-based enrollment.

If you enable the policy and configure it correctly, it first searches for the user under the *base* specified in the `ldap.ldapconn.basedn` parameter with the filter (`uid=HTTP_PARAMS.UID`) for the user's entry.

- If the `value` parameter is empty, the policy checks the `attribute` parameter:
 - If the attribute named in the `attribute` parameter is present in the request, the policy accepts the request.
 - If the attribute named in the `attribute` parameter is not present in the request, the policy rejects the request.
- If the `value` parameter is not empty, the policy checks its value with the value of the attribute specified in the `attribute` parameter.
 - If the attribute named in the `attribute` parameter has the specified value, the policy accepts the request.
 - If the attribute named in the `attribute` parameter does not have the specified value, the policy rejects the request.

In the case of multi-valued attributes, the request will be accepted if any of the values matches the specified value; comparisons are case sensitive.

Unlike some of the other policy modules, Certificate Management System does not create an instance of the attribute present constraints policy during installation. If you want to create an instance of the `AttributePresentConstraints` module, see section “Step 4. Add New Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*.

Configuration Parameters of AttributePresentConstraints

In the CMS configuration file, the `AttributePresentConstraints` module is identified as `<subsystem>.Policy.impl.AttributePresentConstraints.class=com.ipanet.certsrv.policy.AttributePresentConstraints`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `AttributePresentConstraints`. Figure 3-2 shows how configurable parameters for the module are displayed in the CMS window.

Figure 3-2 Parameters of the AttributePresentConstraints module

The screenshot shows the 'Policy Rule Editor' window. At the top, 'Policy Rule ID' is 'PinCheckForClientCerts' and 'Policy Plugin ID' is 'AttributePresentConstraints'. The 'enable' checkbox is checked. The 'predicate' field contains 'HTTP_PARAMS.certType==client'. The LDAP connection parameters are: '* ldap.ldapconn.host' is 'corDirectory.siroe.com', '* ldap.ldapconn.port' is '389', 'ldap.ldapconn.secureConn' is unchecked, '* ldap.ldapconn.version' is '3', 'ldap.ldapauth.bindDN' is 'CN=pinmanager', 'password' is masked with asterisks, 'ldap.ldapauth.clientCertNickname' is empty, '* ldap.ldapauth.authType' is 'Basic:Auth', '* ldap.ldapconn.basedn' is 'O=siroe.com', 'ldap.ldapconn.minConns' is '1', 'ldap.ldapconn.maxConns' is '5', '* attribute' is 'pin', and 'value' is empty. A text area at the bottom says 'Ldap attribute to check presence of (default pin)'. At the bottom are 'OK', 'Cancel', and 'Help' buttons.

The configuration shown in Figure 3-2 creates a policy rule named `PinCheckForClientCerts`, which enforces a rule that the server should check for users PINs in the specified LDAP directory.

Table 3-2 describes each of the parameters.

Table 3-2 Description of parameters defined in the AttributePresentConstraints module

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server applies the rule to certificates specified by the predicate expression.• If you disable the rule, the server does not apply the rule to certificates.

Table 3-2 Description of parameters defined in the AttributePresentConstraints module (*Continued*)

Parameter	Description
<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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client</code></p>
<code>ldap.ldapconn. host</code>	<p>Specifies the host name of the LDAP directory to connect to.</p> <p>Permissible values: The name must be fully-qualified host name in the <code><machine_name>.<your_domain>.<domain></code> form.</p> <p>Example: <code>corpDirectory.siroe.com</code></p>
<code>ldap.ldapconn. port</code>	<p>Specifies the TCP/IP port at which the LDAP directory listens to requests from Certificate Management System.</p> <p>Permissible values: Any valid port number. The default is 389; use 636 if the directory is configured for SSL-enabled communication.</p> <p>Example: <code>389</code></p>
<code>ldap.ldapconn. secureConn</code>	<p>Specifies the type—SSL or non-SSL—of the port at which the LDAP directory listens to requests from Certificate Management System.</p> <ul style="list-style-type: none"> Check the box if the port is an SSL (HTTPS) port. If your directory is configured for SSL-enabled communication (with or without SSL client authentication), choose this option. Leave the box unchecked if the port is a non-SSL (HTTP) port. If your directory is configured for basic authentication, choose this option (default).
<code>ldap.ldapconn. version</code>	<p>Specifies the LDAP protocol version.</p> <p>Permissible values: 2 or 3.</p> <ul style="list-style-type: none"> 2 specifies LDAP version 2. If your directory is based on Netscape Directory Server 1.x, choose 2. 3 specifies LDAP version 3. For Directory Server versions 3.x and later, choose 3 (default). <p>Example: <code>3</code></p>
<code>ldap.ldapauth. bindDN</code>	<p>Specifies the user entry to bind as for checking the attribute in the LDAP directory.</p> <p>Permissible values: A valid bind DN.</p> <p>Example: <code>CN=pinmanager</code></p>
<code>password</code>	<p>Specifies the password associated with the DN specified by the <code>ldap.ldapauthbindDN</code> parameter.</p>

Table 3-2 Description of parameters defined in the AttributePresentConstraints module *(Continued)*

Parameter	Description
<code>ldap.ldapauth.clientCertNickname</code>	<p>Specifies the nickname or the friendly name of the certificate to be used for SSL client authentication to the LDAP directory in order to check attributes. Make sure that the certificate is valid and has been signed by a CA that is trusted in the directory's certificate database, and that the directory's <code>certmap.conf</code> file has been configured to correctly map the certificate to a DN in the directory. (This is needed for PIN removal only.)</p> <p>Permissible values: Enter the name of a currently valid CMS certificate, for example, its SSL server certificate.</p> <p>Example: <code>Server-Cert</code></p>
<code>ldap.ldapauth.authtype</code>	<p>Specifies how to bind to the directory or the authentication type—basic authentication or SSL client authentication—required in order to check attributes in the LDAP directory.</p> <p>Permissible values: <code>BasicAuth</code> or <code>SslClientAuth</code>.</p> <ul style="list-style-type: none"> • <code>BasicAuth</code> specifies basic authentication (default). If you choose this option, be sure to enter the correct values for <code>ldap.ldapauth.bindDN</code> and <code>password</code> parameters; the plug-in uses the DN from the <code>ldap.ldapauth.bindDN</code> attribute to bind to the directory. • <code>SslClientAuth</code> specifies SSL client authentication. If you choose this option, be sure to select the <code>ldap.ldapconn.secureConn</code> parameter and set the value of the <code>ldap.ldapauth.clientCertNickname</code> parameter to the nickname of the certificate to be used for SSL client authentication. <p>Example: <code>BasicAuth</code></p>
<code>ldap.ldapconn.binddn</code>	<p>Specifies the base DN for searching the LDAP directory—the plug-in uses the value of the <code>uid</code> field from the HTTP input (what a user enters in the enrollment form) and the base DN to construct an LDAP search filter.</p> <p>Permissible values: Any valid DN string of up to 255 characters. (If your user's DN is <code>uid=jdoe, o=company</code>, you might want to use <code>o=company</code> here.)</p> <p>Example: <code>O=siroe.com</code></p>
<code>ldap.ldapconn.minConns</code>	<p>Specifies the minimum number of connections permitted (or to keep open) to the LDAP directory.</p> <p>Permissible values: 1 to 3; the default value is 1.</p> <p>Example: 3</p>

Table 3-2 Description of parameters defined in the AttributePresentConstraints module (*Continued*)

Parameter	Description
<code>ldap.ldapconn.maxConns</code>	Specifies the maximum number of connections permitted to the LDAP directory; when needed, connection pool can grow to this many (multiplexed) connections. Permissible values: 3 to 10; the default value is 5. Example: 9
<code>attribute</code>	Specifies the LDAP attribute, the presence of which is to be checked in the certificate-enrollment request. Permissible values: Valid directory attributes, separated by commas; the default value is <code>pin</code> . Example: <code>pin</code>
<code>value</code>	If this parameter is non-empty, the attribute value must match this value for the request to proceed to the next stage.

DSAKeyConstraints Plug-in Module

The `DSAKeyConstraints` plug-in module implements the DSA key constraints policy. This policy imposes constraints on the following:

- The minimum and maximum sizes for keys
- The sizes of exponents

The policy restricts the key size to one of the sizes, such as 512 or 1024, supported by Certificate Management System.

You may apply this policy to end-entity certificate enrollment and renewal requests. For example, if you want your CA to certify public keys up to 512 bits in length for end users and 1024 for servers, you can configure Certificate Management System to do so using the policy.

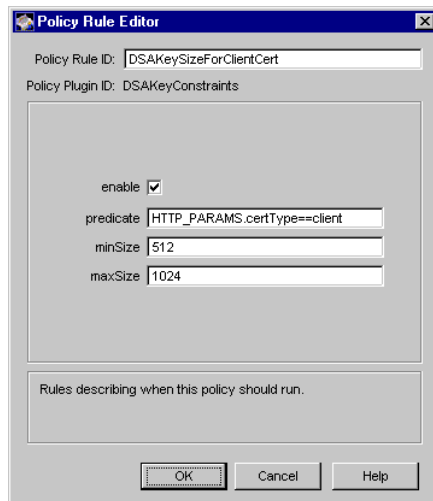
During installation, Certificate Management System automatically creates an instance of the DSA key constraints policy. See “DSAKeyRule Rule” on page 98.

Configuration Parameters of DSAKeyConstraints

In the CMS configuration file, the `DSAKeyConstraints` module is identified as `<subsystem>.Policy.impl.DSAKeyConstraints.class=com.iplanet.certsrv.policy.DSAKeyConstraints`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `DSAKeyConstraints`. Figure 3-3 shows how configurable parameters for the module are displayed in the CMS window.

Figure 3-3 Parameters of the `DSAKeyConstraints` module



The configuration shown in Figure 3-3 creates a policy rule named `DSAKeySizeForClientCert`, which enforces a rule that the server should restrict the minimum and maximum key sizes for all DSA key-based client certificates to 512 and 1024, respectively.

Table 3-3 describes each of the parameters.

Table 3-3 Description of parameters defined in the DSAKeyConstraints module

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none"> If you enable the rule and set the remaining parameters correctly, the server applies the rule to certificates specified by the predicate expression. If you disable the rule, the server does not apply the rule to certificates.
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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client</code></p>
minSize	<p>Specifies the minimum length, in bits, for the key (the length of the modulus in bits). The value must be smaller than or equal to the one specified by the <code>maxSize</code> parameter.</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>Permissible values: 512 or 1024. You may also enter a custom key size that is between 512 and 1024, in increments of 64 bits. The default value is 512.</p> <p>Example: 512</p>
maxSize	<p>Specifies the maximum length, in bits, for the key.</p> <p>Permissible values: 512 or 1024. You may also enter a custom key size that is between 512 and 1024, in increments of 64 bits. The default value is 1024.</p> <p>Example: 1024</p>
exponents	<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>Permissible values: A combination of 3, 7, 17, and 65537, separated by commas. The default value is 3,7,17,65537.</p> <p>Example: 17,65537</p>

DSAKeyRule Rule

The rule named `DSAKeyRule` is an instance of the `DSAKeyConstraints` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression is left blank so that the rule is applied to all certificate enrollment and renewal requests processed by the server.
- The minimum key size permitted for certificates is 512 bits (`minSize=512`).
- The maximum key size permitted for certificates is 1024 bits (`maxSize=1024`).
- The exponents allowed are 3, 7, 17, and 65537 (`exponents=3,7,17,65537`).

For details on individual parameters defined in the rule, see Table 3-3 on page 97. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

IssuerConstraints Plug-in Module

The `IssuerConstraints` plug-in module implements the issuer constraints policy. The policy enables you to effectively deploy certificate-based enrollment explained in “Certificate-Based Enrollment” on page 53.

The policy enables the Certificate Manager to authenticate an end user by checking the issuer DN of the CA that has issued the certificate the user presents as an enrollment token during enrollment. Note that in the current implementation, the CA that issues the new certificates must be the same as the one that has issued the certificates used for SSL client authentication; that is, the issuer DN in the authentication certificate must match the issuer DN specified in the policy configuration.

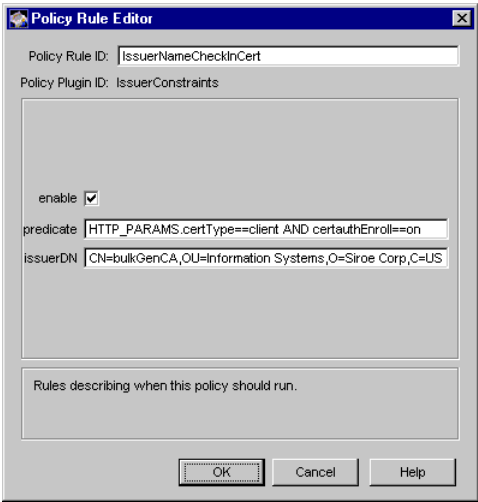
During installation, Certificate Management System automatically creates an instance of the issuer constraints policy. See “IssuerRule Rule” on page 100. The server also provides appropriate enrollment forms for the three certificate-based enrollment scenarios explained above; see “Enrollment Forms” on page 57.

Configuration Parameters of IssuerConstraints

In the CMS configuration file, the IssuerConstraints module is identified as `ca.Policy.impl.IssuerConstraints.class=com.iplanet.certsrv.policy.IssuerConstraints`.

In the CMS window, the module is identified as `IssuerConstraints`. Figure 3-4 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 3-4 Parameters of the IssuerConstraints module



The configuration shown in Figure 3-4 creates a policy rule named `IssuerNameCheckInCert`, which enforces a rule that the server should check for certificates issued by a CA, whose issuer DN is `CN=bulkGenCA,OU=Information Systems,O=Siroe Corp,C=US`.

Table 3-4 describes each parameter.

Table 3-4 Description of parameters defined in the IssuerConstraints module

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none"> If you enable the rule and set the remaining parameters correctly, the server checks for certificates issued by the specified CA and enforces certificate-based enrollment. If you disable the rule, the server does not check for certificates issued by a CA; it ignores the values specified in the remaining fields.
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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client AND certauthEnroll==on</code></p>
issuerDN	<p>Specifies the name of the CA that has issued certificates that are to be checked. You should enter the issuer name as it appears in the CA’s signing certificate; the same name also appears as the issuer name in certificates the CA signs.</p> <p>Permissible values: A valid issuer name.</p> <p>Example: <code>CN=bulkGenCA,OU=Information Systems,O=Siroe Corp,C=US</code></p>

IssuerRule Rule

The rule named `IssuerRule` is an instance of the `IssuerConstraints` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is disabled; for the rule to be effective, it must be enabled and configured appropriately.
- The predicate expression is set (`predicate=HTTP_PARAMS.certType==client AND certauthEnroll==on`) so that the rule is applied to only those client-certificate requests that have certificate-based authentication turned on.
- The issuer DN field is left blank for you to enter the appropriate information.

For details on individual parameters defined in the rule, see Table 3-4 on page 100. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

KeyAlgorithmConstraints Plug-in Module

The `KeyAlgorithmConstraints` plug-in module implements the key algorithm constraints policy. This policy restricts the key algorithm requested in certificates to the algorithms, such as RSA and 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.

You may apply this policy to end-entity certificate enrollment and renewal requests. For example, if you want your CA to certify only those public keys that comply with the PKCS-1 RSA Encryption Standard, you can configure the server for that using the policy.

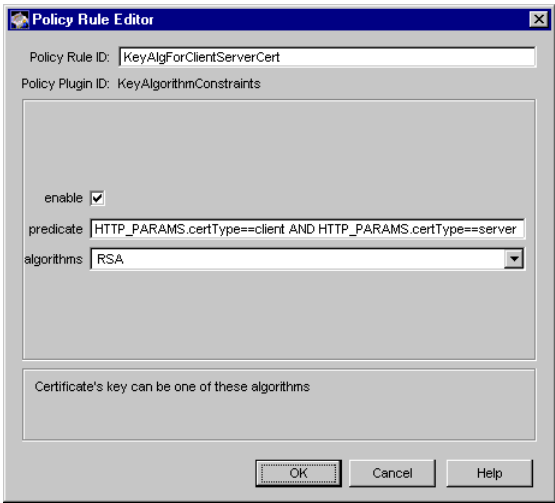
During installation, Certificate Management System automatically creates an instance of the key algorithm constraints policy. See “KeyAlgRule Rule” on page 103.

Configuration Parameters of KeyAlgorithmConstraints

In the CMS configuration file, the `KeyAlgorithmConstraints` module is identified as `<subsystem>.Policy.impl.KeyAlgorithmConstraints.class=com.iplanet.certsrv.policy.KeyAlgorithmConstraints`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `KeyAlgorithmConstraints`. Figure 3-5 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 3-5 Parameters of the KeyAlgorithmConstraints module



The configuration shown in Figure 3-5 creates a policy rule named `KeyAlgForClientServerCert`, which enforces a rule that the server should restrict the key algorithm of all client and server certificates to RSA.

Table 3-5 gives details about each of the parameters.

Table 3-5 Description of parameters defined in the KeyAlgorithmConstraints module

Parameter	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server sets the configured algorithm in certificates specified by the <code>predicate</code> parameter.• If you disable the rule, the server sets the algorithm specified in the certificate request.
<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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client AND HTTP_PARAMS.certType==server</code></p>

Table 3-5 Description of parameters defined in the KeyAlgorithmConstraints module (*Continued*)

Parameter	Description
algorithms	<p>Specifies the key type the server should certify. The default is RSA.</p> <p>Permissible values: RSA, DSA, or RSA, DSA.</p> <p>Example: RSA</p>

KeyAlgRule Rule

The rule named `KeyAlgRule` is an instance of the `KeyAlgorithmConstraints` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression is left blank so that the rule is applied to all certificate enrollment and renewal requests processed by the server.
- The key type allowed is RSA (`algorithms=RSA`).

For details on individual parameters defined in the rule, see Table 3-5 on page 102. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section F“Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

RenewalConstraints Plug-in Module

The `RenewalConstraints` plug-in module implements the renewal constraints policy. This policy imposes constraints on renewal of expired certificates—it allows or restricts the server from renewing expired certificates. You may apply this policy to end-entity certificate renewal requests. For example, if you don’t want to allow renewal of expired certificates, you can configure the server accordingly using the policy.

In certain situations you may want to allow renewal of expired certificates. Here’s one such scenario: the renewal validity constraints policy (see “RenewalValidityConstraints Plug-in Module” on page 106) allows you to delay renewal of certificates as long as possible to reduce the overhead of processing new certificate requests. Typically, you would limit the renewal process to the last few

weeks of validity of the certificate. However, if the interval specified in the policy rule is not sufficient for renewal to occur, some of your users may not be able to renew their certificates prior to the expiration time and end up owning expired certificates.

Note that the policy also allows you to specify how long after the expiration of a certificate can it be renewed. If you don't specify this, the server will renew all expired certificates that are submitted for renewal.

During installation, Certificate Management System automatically creates an instance of the renewal constraints policy. See “RenewalConstraintsRule Rule” on page 105.

Configuration Parameters of RenewalConstraints

In the CMS configuration file, the `RenewalConstraints` module is identified as `<subsystem>.Policy.impl.RenewalConstraints.class=com.iplanet.certsrv.policy.RenewalConstraints`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `RenewalConstraints`. Figure 3-6 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 3-6 Parameters of the RenewalConstraints module



The configuration shown in Figure 3-6 creates a policy rule named `RenewExpiredClientCert`, which specifies that the server should allow renewal of expired client certificates, if it's done within 30 days from the expiry date.

Table 3-6 gives details about each of the parameters.

Table 3-6 Description of parameters defined in the `RenewalConstraints` module

Parameter	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none"> If you enable the rule and set the remaining parameters correctly, the server verifies the validity period of the certificate being renewed, checks the value assigned to the <code>allowExpiredCerts</code> parameter, and accordingly allows or denies the renewal request. If you disable the rule, the server does not verify the validity period of the certificate being renewed; it simply renews the certificate.
<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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client</code></p>
<code>allowExpiredCerts</code>	<p>Specifies whether to allow or prevent renewal of expired certificates. Check the box if you want the server to renew expired certificates (default). Uncheck the box if you don't want the server to renew expired certificates.</p>
<code>renewalNotAfter</code>	<p>Specifies how long, in days, after the expiration of a certificate can it be renewed. The default value is 30 days. If you leave the field blank, the server will renew all expired certificates that are submitted for renewal.</p> <p>Example: 15</p>

RenewalConstraintsRule Rule

The rule named `RenewalConstraintsRule` is an instance of the `RenewalConstraints` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression is left blank so that the policy is applied to all certificate renewal requests processed by the server.

- The server allows renewal of expired certificates within 30 days, starting from the date they expire.

For details on individual parameters defined in the rule, see Table 3-6 on page 105. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

RenewalValidityConstraints Plug-in Module

The `RenewalValidityConstraints` plug-in module implements the renewal validity constraints policy. This policy governs the formulation of content in the renewed certificate based on the currently issued certificate.

Every certificate issued by Certificate Management System is valid for a limited duration, which is determined by the validity period specified in the validity constraints policy (see “ValidityConstraints Plug-in Module” on page 124) at the time the certificate is issued. In order to continue to participate in the PKI-using system beyond this validity period, the entity owning the certificate must renew the certificate; the new certificate generally contains a new validity time period and some updated attributes.

To eliminate administrative overhead of monitoring certificate validity periods and reminding users to renew their certificates, Certificate Management System provides a schedulable job that can detect any to-be-expired certificates and automatically remind users to renew their certificates. For details about this job, see “RenewalNotificationJob Plug-in Module” on page 69.

The renewal validity constraints policy enables you to enforce certain restrictions on certificate-renewal requests, when end entities attempt to renew their certificates. You can specify restrictions on the following:

- The number of days before expiration that end entities can renew their currently active or valid certificates. For example, if you want to prevent end entities from renewing their certificates any earlier than 15 days before expiration, you can configure the server accordingly using the policy.
- The validity period of the renewed certificate. For example, if you want the validity period of all renewed certificates to be a minimum of 180 days, you can configure the server accordingly using the policy. Note that the renewal period starts from the ending period in the certificate presented for renewal.

Note that you may apply this policy to certificate renewal requests only, and the renewal process to which this policy is applied can be manual (a request needs to be approved by an agent) or automated. 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.

By default, any validity requested in a certificate-renewal request cannot exceed beyond that of the expiration time specified in the CA's signing certificate (see section "CA Signing Key Pair Certificate" in Chapter 14, "Managing CMS Keys and Certificates" of *CMS Installation and Setup Guide*). If the Certificate Manager (CA) finds a request with validity period extending beyond that of its CA signing certificate, it automatically truncates the validity period to end on the day the CA signing certificate expires. For example, if the CA signing certificate expires on June 10, 2004, any renewal request with validity period beyond June 10, 2004 will have validity period truncated to end on June 10, 2004.

However, you can configure the Certificate Manager to renew certificates with validity periods beyond that of its CA signing certificate by selecting the "Override validity nesting requirement" option; see section F"Step 6. Enable End-Entity Interaction" in Chapter 15, "Setting Up End-User Authentication" of *CMS Installation and Setup Guide*.

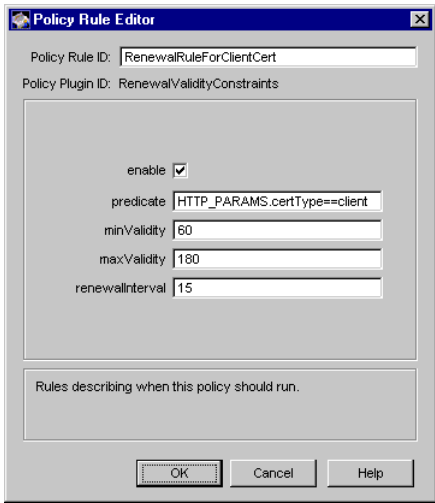
During installation, Certificate Management System automatically creates an instance of the renewal validity constraints policy. See "DefaultRenewalValidityRule Rule" on page 109.

Configuration Parameters of RenewalValidityConstraints

In the CMS configuration file, the `RenewalValidityConstraints` module is identified as `<subsystem>.Policy.impl.RenewalValidityConstraints.class=com.ipplanet.certsrv.policy.RenewalValidityConstraints`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `RenewalValidityConstraints`. Figure 3-7 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 3-7 Parameters of the RenewalValidityConstraints module



The configuration shown in Figure 3-7 creates a policy rule named `RenewalRuleForClientCert`, which enforces a rule that the server should renew only those client certificates that are due to expire within the next 15 days. The renewed certificates are valid for at least 60 days (two months) and require renewing after 180 days (six months).

Table 3-7 gives details about each of the parameters.

Table 3-7 Description of parameters defined in the RenewalValidityConstraints module

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server sets the configured validity period in renewed certificates specified by the <code>predicate</code> parameter.• If you disable the rule, the server sets the validity period as specified in the renewal request.
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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client</code></p>

Table 3-7 Description of parameters defined in the RenewalValidityConstraints module *(Continued)*

Parameter	Description
minValidity	Specifies the minimum validity period, in days, for renewed certificates. Permissible values: As applicable. The default value is 180 days. Example: 60
maxValidity	Specifies the maximum validity period, in days, for renewed certificates. Permissible values: As applicable. The default value is 730 days. Example: 180
renewalInterval	Specifies how many days before its expiration that a certificate can be renewed. Permissible values: As applicable. The default value is 15 days. Example: 15

DefaultRenewalValidityRule Rule

The rule named `DefaultRenewalValidityRule` is an instance of the `RenewalValidityConstraints` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression is left blank so that the policy is applied to all certificate renewal requests processed by the server.
- The minimum validity period permitted for renewed certificates is 30 days (`minValidity=30`).
- The maximum validity period permitted for renewed certificates is 365 days (`maxValidity=365`).
- The number of days before expiration that end entities can renew their currently valid certificates is 15 (`renewalInterval=15`).

For details on individual parameters defined in the rule, see Table 3-7 on page 108. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

RevocationConstraints Plug-in Module

The `RevocationConstraints` plug-in module implements the revocation constraints policy. This policy imposes constraints on revocation of expired certificates—it allows or restricts the server from revoking expired certificates. You may apply this policy to end-entity certificate revocation requests. For example, if you don't want to allow revocation of expired certificates in your PKI setup, you can configure the server accordingly using the policy.

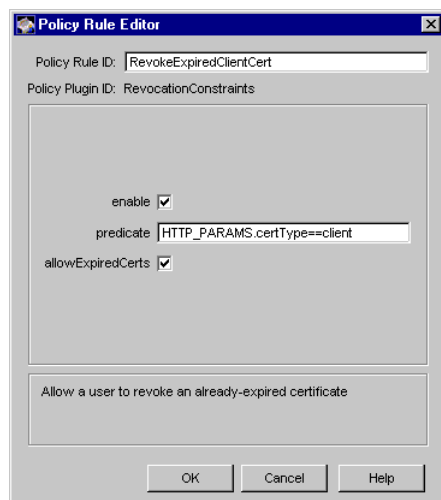
During installation, Certificate Management System automatically creates an instance of the revocation constraints policy. See “Configuration Parameters of `RevocationConstraints`” on page 110.

Configuration Parameters of `RevocationConstraints`

In the CMS configuration file, the `RevocationConstraints` module is identified as `<subsystem>.Policy.impl.RevocationConstraints.class=com.iplanet.certsrv.policy.RevocationConstraints`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `RevocationConstraints`. Figure 3-8 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 3-8 Parameters of the `RevocationConstraints` module



The configuration shown in Figure 3-8 creates a policy rule named `RevokeExpiredClientCert`, which specifies that the server should allow revocation of expired client certificates.

Table 3-8 gives details about each of the parameters.

Table 3-8 Description of parameters defined in the `RevocationConstraints` module

Parameter	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server verifies the validity period of the certificate being revoked, checks the value assigned to the <code>allowExpiredCerts</code> parameter, and accordingly allows or denies the revocation request.• If you disable the rule, the server does not verify the validity period of the certificate being revoked; it simply revokes the certificate.
<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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client</code></p>
<code>allowExpiredCerts</code>	<p>Specifies whether to allow or prevent revocation of expired certificates. Check the box if you want the server to revoke expired certificates (default). Uncheck the box if you don’t want the server to revoke expired certificates.</p>

RevocationConstraintsRule Rule

The rule named `RevocationConstraintsRule` is an instance of the `RevocationConstraints` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression is left blank so that the policy is applied to all certificate revocation requests processed by the server.
- The server allows revocation of expired certificates.

For details on individual parameters defined in the rule, see Table 3-8 on page 111. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

RSAKeyConstraints Plug-in Module

The `RSAKeyConstraints` plug-in module implements the RSA key constraints policy. This 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 supported by Certificate Management System—512, 1024, 2048, or 4096. In other words, the policy allows you to set up restrictions on the lengths of public keys certified by Certificate Management System.

You may apply this policy to end-entity certificate enrollment and renewal requests. For example, if you want your CA to certify public keys up to 1024 bits in length for end users, you can configure the server accordingly using the policy.

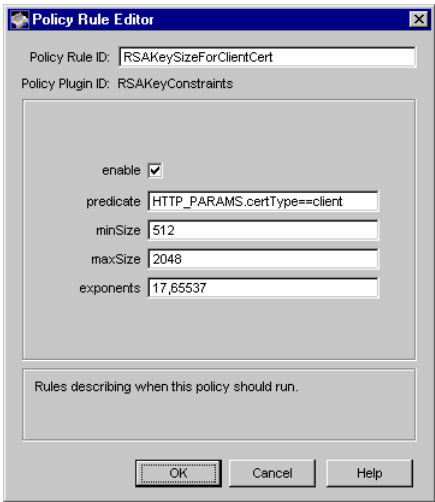
During installation, Certificate Management System automatically creates an instance of the RSA key constraints policy. See “RSAKeyRule Rule” on page 114.

Configuration Parameters of RSAKeyConstraints

In the CMS configuration file, the `RSAKeyConstraints` module is identified as `<subsystem>.Policy.impl.RSAKeyConstraints.class=com.iplanet.certsrv.policy.RSAKeyConstraints`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `RSAKeyConstraints`. Figure 3-9 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 3-9 Parameters of the RSACKeyConstraints module



The configuration shown in Figure 3-9 creates a policy rule named `RSACKeySizeForClientCert`, which enforces a rule that the server should restrict the minimum and maximum key sizes for all RSA key-based client certificates to 512 and 2048, respectively.

Table 3-9 describes each parameter.

Table 3-9 Description of parameters defined in the RSACKeyConstraints module

Parameter	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server uses the configured RSA key rules when issuing certificates specified by the <code>predicate</code> parameter.• If you disable the rule, the server certifies the requested key size.
<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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client</code></p>

Table 3-9 Description of parameters defined in the RSAKeyConstraints module *(Continued)*

Parameter	Description
<code>minSize</code>	<p>Specifies the minimum length, in bits, for the key (the length of the modulus in bits). The value must be smaller than or equal to the one specified by the <code>maxSize</code> parameter.</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>Permissible values: 512, 1024, 2048, or 4096. You may also enter a custom key size that is between 512 and 4096 bits. The default value is 512.</p> <p>Example: 512</p>
<code>maxSize</code>	<p>Specifies the maximum length, in bits, for the key.</p> <p>Permissible values: 512, 1024, 2048, or 4096. You may also enter a custom key size that is between 512 and 4096 bits. The default value is 2048.</p> <p>Example: 1024</p>
<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>Permissible values: A combination of 3, 7, 17, and 65537, separated by commas. The default value is 3,7,17,65537.</p> <p>Example: 17,65537</p>

RSAKeyRule Rule

The rule named `RSAKeyRule` is an instance of the `RSAKeyConstraints` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is disabled; for the rule to be effective, it must be enabled and configured appropriately.
- The predicate expression is left blank so that the rule is applied to all certificate enrollment and renewal requests processed by the server.
- The minimum key size permitted for certificates is 512 bits (`minSize=512`).

- The maximum key size permitted for certificates is 2048 bits (`maxSize=2048`).
- The exponents allowed are 3, 7, 17, and 65537 (`exponents=3,7,17,65537`).

For details on individual parameters defined in the rule, see Table 3-9 on page 113. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

SigningAlgorithmConstraints Plug-in Module

The `SigningAlgorithmConstraints` plug-in module implements the signing algorithm constraints policy. This policy restricts the requested signing algorithm to be one of the algorithms supported by Certificate Management System: MD2 with RSA, MD5 with RSA, and SHA-1 with RSA, if the Certificate Manager’s signing key is RSA and SHA-1 with DSA, if the Certificate Manager’s signing key is DSA.

When a Certificate Manager digitally signs a message, it generates a compressed version of the message called a message digest. Some of the algorithms used to produce this digest include MD5 and SHA-1 (Secure Hash Algorithm).

- MD5 generates a 128-bit message digest. Most existing software applications that handle certificates only support MD5.
- SHA-1 generates a 160-bit message digest. Some software applications do not yet support the SHA-1 algorithm. For example, Netscape Navigator 3.0 (or higher) and Enterprise Server 2.01 (or higher) support SHA-1; previous versions of these applications do not support SHA-1.

You may apply this policy to end-entity certificate enrollment and renewal requests.

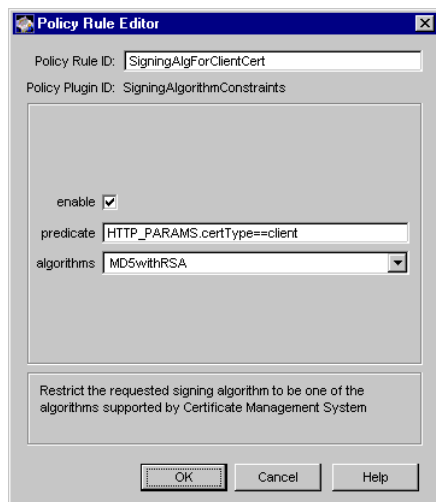
During installation, Certificate Management System automatically creates an instance of the signing algorithm constraints policy. See “SigningAlgRule Rule” on page 118.

Configuration Parameters of SigningAlgorithmConstraints

In the CMS configuration file, the `SigningAlgorithmConstraints` module is identified as `<subsystem>.Policy.impl.SigningAlgorithmConstraints.class=com.iplanet.certsrv.policy.SigningAlgorithmConstraints`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `SigningAlgorithmConstraints`. Figure 3-10 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 3-10 Parameters of the `SigningAlgorithmConstraints` module



The configuration shown in Figure 3-10 creates a policy rule named `SigningAlgForClientCert`, which enforces a rule that the server should use MD5 with RSA signature algorithm to sign all client certificates.

Table 3-10 provides details for each of these parameters.

Table 3-10 Description of parameters defined in the SigningAlgorithmConstraints module

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none"> If you enable the rule and set the remaining parameters correctly, the server uses the configured algorithms to sign certificates specified by the <code>predicate</code> parameter. If you disable the rule, the server uses the default algorithm specified for the Certificate Manager; see Certificate Manager's "General Settings" tab in the CMS window.
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 (default). To form a predicate expression, see section "Using Predicates in Policy Rules" in Chapter 18, "Setting Up Policies" of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client</code></p>
algorithms	<p>Specifies the signature algorithm the server should use to sign certificates.</p> <p>Permissible values: Depends on the CA's signing key type (the key type you chose for the Certificate Manager's CA signing certificate).</p> <ul style="list-style-type: none"> If the key type is RSA, select one of the following: <ul style="list-style-type: none"> MD2withRSA, MD5withRSA, SHA1withRSA MD2withRSA, MD5withRSA MD2withRSA, SHA1withRSA MD5withRSA, SHA1withRSA MD2withRSA MD5withRSA SHA1withRSA <p>The default value is MD2withRSA, MD5withRSA, SHA1withRSA.</p> If the key type is DSA, select SHA1withDSA. <p>Example: <code>MD5withRSA</code></p>

SigningAlgRule Rule

The rule named `SigningAlgRule` is an instance of the `SigningAlgorithmConstraints` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression is left blank so that the rule is applied to all certificate enrollment and renewal requests processed by the server.
- The signature algorithms allowed are MD5 with RSA, MD2 with RSA, and SHA-1 with RSA (`algorithms=MD5withRSA,MD2withRSA,SHA1withRSA`).

For details on individual parameters defined in the rule, see Table 3-10 on page 117. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

SubCANameConstraints Plug-in Module

The `SubCANameConstraints` plug-in module implements the subordinate CA name constraints policy. This policy restricts a CA from issuing a subordinate CA certificate that has the same issuer name as that of the CA itself—that is, the policy prevents a situation where the signing certificates of a CA and its subordinate CA have identical issuer names.

This policy must be turned on if you’re planning to issue subordinate CA certificates. The reason for this is that, whenever the Certificate Manager issues a certificate, it stores the related information in its internal database; see Chapter 12, “Setting Up Internal Database” of *CMS Installation and Setup Guide*. If the CA issues a subordinate CA certificate with an issuer DN that matches its own issuer DN, the internal database will not function properly.

You may apply this policy to CA certificate enrollment and renewal requests.

During installation, Certificate Management System automatically creates an instance of the subordinate CA name constraints policy. See “SubCANameConstraints Rule” on page 120.

Configuration Parameters of SubCANameConstraints

In the CMS configuration file, the `SubCANameConstraints` module is identified as `ca.Policy.impl.SubCANameConstraints.class=`
`com.ipplanet.certsrv.policy.SubCANameConstraints`.

In the CMS window, the module is identified as `SubCANameConstraints`. Figure 3-11 shows how configurable parameters for the module are displayed in the CMS window.

Figure 3-11 Parameters of the SubCANameConstraints module



The configuration shown in Figure 3-11 creates a policy rule named `IssuerDNCheckForSubCACert`, which enforces a rule that the server should reject subordinate CA certificate requests with issuer DNs matching its own issuer DN.

Table 3-11 gives details about each of the parameters.

Table 3-11 Description of parameters defined in the SubCANameConstraints module

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule. Uncheck the box to disable the rule (default).</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server checks the certificate requests for issuer name uniqueness. If a certificate with the requested issuer name already exists in the internal database, the server rejects the request.• If you disable the rule, the server does not check the CA certificate requests for issuer name uniqueness.
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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==ca</code></p>

SubCANameConstraints Rule

The rule named `SubCANameConstraints` is an instance of the `SubCANameConstraints` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression is left blank so that the rule is applied to all certificate enrollment and renewal requests processed by the server.

For details on individual parameters defined in the rule, see Table 3-12 on page 122. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section F“Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

UniqueSubjectNameConstraints Plug-in Module

The `UniqueSubjectNameConstraints` plug-in module implements the unique subject name constraints policy. This policy restricts the server from issuing multiple certificates with same subject names. Optionally, you can also configure the server to allow multiple certificates with the same subject name if the key usages are different. Note that key usages for certificates are usually specified by the *key usage extension* and Certificate Management System allows you to add this extension to certificates using the key usage extension policy explained in “KeyUsageExt Plug-in Module” on page 189.

You may apply the unique subject name constraints policy to end-entity certificate enrollment and renewal requests. For example, if you want to prevent your users from requesting multiple certificates with same subject names, you can configure the server accordingly using the policy. Alternatively, if you want to allow your users to own multiple certificates, each for a different use, all having the same subject name, you can do so easily using the `enableKeyUsageExtensionChecking` parameter defined in this policy. This parameter makes the server check whether the key usages specified in the certificate request being processed is different than those specified in the existing certificates that have the same subject names and accordingly issue or deny the certificate. Keep in mind that the server can check for key usages only if the key usage extension bits are set in the certificate request being processed as well as in the existing certificates that have the same subject names.

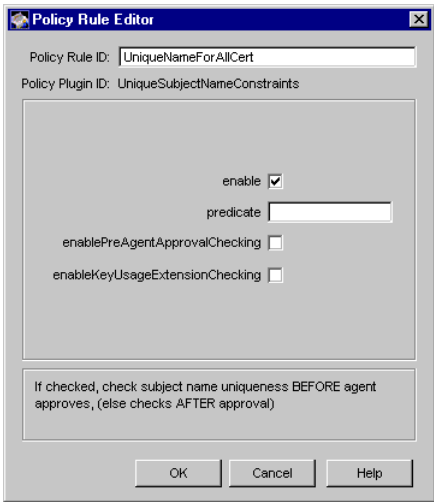
During installation, Certificate Management System automatically creates an instance of the unique subject name constraints policy. See “UniqueSubjectNameConstraints Rule” on page 124.

Configuration Parameters of UniqueSubjectNameConstraints

In the CMS configuration file, the `UniqueSubjectNameConstraints` module is identified as `ca.Policy.impl.UniqueSubjectNameConstraints.class=com.iplanet.certsrv.policy.UniqueSubjectNameConstraints`.

In the CMS window, the module is identified as `UniqueSubjectNameConstraints`. Figure 3-12 shows how configurable parameters for the module are displayed in the CMS window.

Figure 3-12 Parameters of the UniqueSubjectNameConstraints module



The configuration shown in Figure 3-12 creates a policy rule named UniqueNameForAllCert, which enforces a rule that all certificates must have unique subject names.

Table 3-12 describes each of the parameters.

Table 3-12 Description of parameters defined in the UniqueSubjectNameConstraints module

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule. Uncheck the box to disable the rule (default).</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server checks the certificate requests, as specified by the predicate parameter, for subject name uniqueness.• If you disable the rule, the server does not check the certificate requests for subject name uniqueness.
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 (default). To form a predicate expression, see section F“Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client</code></p>

Table 3-12 Description of parameters defined in the UniqueSubjectNameConstraints module (*Continued*)

Parameter	Description
enablePreAgentApprovalChecking	<p>Specifies whether the request must be checked for the subject name uniqueness on submission by the user, before the request gets queued for agent approval.</p> <ul style="list-style-type: none"> • Check the box if you want the server to check the certificate request for the subject name uniqueness as soon as the user submits it. • Uncheck the box if you want the server to check the certificate request for the subject name uniqueness after agent approval; that is, you want the policy to be applied to the request after an agent approves the request. You should choose this option if you want the server to check the Key Usage extension (see “KeyUsageExt Plug-in Module” on page 189) before determining whether to issue the certificate.
enableKeyUsageExtensionChecking	<p>Specifies whether the certificate request must be checked for the Key Usage extension. Note that the policy can check the certificate request for the Key Usage extension only if you uncheck (disable) the enablePreAgentApprovalChecking parameter. The reason for this is that, extensions are set on the request after agent approval, so this checking can be done after an agent approves the request.</p> <ul style="list-style-type: none"> • Check the box if you want the server to check the certificate request for the Key Usage extension. If you check the box, the server checks its internal database for certificates that have the same subject name as the one specified in the request. For each certificate that has the matching subject name, the server compares the Key Usage extension of the certificate to the one specified in the request. If the server finds a certificate that has the same subject name and Key Usage extension, it rejects request. Otherwise, the server approves the request. (This choice is suitable if you want to have multiple certificates with same subject names but for different purposes, such as signing and encrypting. If key-usage comparison is to be done, be sure to specify that this policy is to be applied <i>after</i> the Key Usage extension policy; see section “Step 5. Reorder Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.) • Uncheck the box if you don’t want the server to check the certificate request for the Key Usage extension. If you uncheck the box, the server does not compare the Key Usage extension in the request with the ones set in the existing certificates that have the same subject name; it simply rejects requests with same subject names.

UniqueSubjectNameConstraints Rule

The rule named `UniqueSubjectNameConstraints` is an instance of the `UniqueSubjectNameConstraints` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is disabled; for the rule to be effective, it must be enabled and configured appropriately.
- The certificate requests are checked for subject name uniqueness after agents process the requests for approval—if you're using manual enrollment and *deferred* requests.
- The certificate requests are checked for Key Usage extension.
- The predicate expression is left blank so that the rule is applied to all certificate enrollment and renewal requests processed by the server.

For details on individual parameters defined in the rule, see Table 3-12 on page 122. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

ValidityConstraints Plug-in Module

The `ValidityConstraints` plug-in module implements the validity constraints policy. This 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 01/15/2000 (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 01/15/2000 (mm/dd/yyyy) and 1:30 p.m., the value of the `notBefore` attribute is set to 1:15 p.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

When applying the validity constraints policy, the server does not check the lag time in all certificate requests. It checks the lag time only in those requests that are based on the CRMF protocol—currently, CRMF is the only enrollment format that allows an end entity to request a specific validity period with the `notBefore` attribute set to a time in the past.

You may apply this policy to end-entity certificate enrollment requests. It can be useful to restrict the length of the validity period for certificates issued by the server. 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.

By default, any validity requested in a certificate enrollment request cannot exceed beyond that of the expiration time specified in the CA's signing certificate. If the Certificate Manager (CA) finds a request with validity period extending beyond that of its CA signing certificate, it automatically truncates the validity period to end on the day the CA signing certificate expires. For example, if the CA signing certificate expires on June 10, 2004, any enrollment request with validity period beyond June 10, 2004 will have validity period truncated to end on June 10, 2004.

However, you can configure the Certificate Manager to issue certificates with validity periods beyond that of its CA signing certificate by selecting the “Override validity nesting requirement” option; see section “Step 6. Enable End-Entity Interaction” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*.

During installation, Certificate Management System automatically creates an instance of the validity constraints policy. See “DefaultValidityRule Rule” on page 128.

Configuration Parameters of ValidityConstraints

In the CMS configuration file, the `ValidityConstraints` module is identified as `<subsystem>.Policy.impl.ValidityConstraints.class=com.ipplanet.certsrv.policy.ValidityConstraints`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `ValidityConstraints`. Figure 3-13 shows how configurable parameters for the module are displayed in the CMS window.

Figure 3-13 Parameters of the ValidityConstraints module

The screenshot shows a window titled "Policy Rule Editor". It contains the following fields and controls:

- Policy Rule ID:** A text field containing "ValidityForClientCert".
- Policy Plugin ID:** A text field containing "ValidityConstraints".
- enable:** A checkbox that is checked.
- predicate:** A text field containing the expression "HTTP_PARAMS.certType==client AND HTTP_PARAMS.OU==Marketing".
- minValidity:** A text field containing "60".
- maxValidity:** A text field containing "180".
- leadTime:** A text field containing "10".
- lagTime:** A text field containing "10".
- notBeforeSkew:** A text field containing "5".
- Rules describing when this policy should run:** A large empty text area.
- Buttons:** "OK", "Cancel", and "Help" buttons at the bottom right.

The configuration shown in Figure 3-13 creates a policy rule named `ValidityForClientCert`, which enforces a rule that all client certificates requested by end users 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 3-13 gives details about each of the parameters.

Table 3-13 Description of parameters defined in the ValidityConstraints module

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none"> If you enable the rule and set the remaining parameters correctly, the server sets the configured validity period in certificates specified by the <code>predicate</code> parameter. If you disable the rule, the server does not set the configured validity period in certificates; it sets the validity period to the one specified in the request.
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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client AND HTTP_PARAMS.OU==Marketing</code></p>
minValidity	<p>Specifies the minimum validity period, in days, for certificates.</p> <p>Permissible values: An integer greater than zero and less than the value specified by the <code>maxValidity</code> parameter. The default value is 180 days.</p> <p>Example: 60</p>
maxValidity	<p>Specifies the maximum validity period, in days, for certificates.</p> <p>Permissible values: An integer greater than zero and also greater than the value specified by the <code>minValidity</code> parameter. The default value is 730 days.</p> <p>Example: 180</p>
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.</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>Permissible values: As applicable. The default value is 10 minutes.</p> <p>Example: 10</p>

Table 3-13 Description of parameters defined in the ValidityConstraints module *(Continued)*

Parameter	Description
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.</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>Permissible values: As applicable. The default value is 10 minutes.</p> <p>Example: 10</p>
notBeforeSkew	<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 download.</p> <p>Permissible values: As applicable. The default value is 5 minutes.</p> <p>Example: 5</p>

DefaultValidityRule Rule

The rule named `DefaultValidityRule` is an instance of the `ValidityConstraints` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression is left blank so that the rule is applied to all certificate enrollment and renewal requests processed by the server.
- The minimum validity period allowed for certificates is 1 day (`minValidity=1`).
- The maximum validity period allowed for certificates is 365 days (`maxValidity=365`).
- The lead time allowed is 10 minutes (`leadTime=10`).
- The lag time allowed is 10 minutes (`lagTime=10`).

- The the number of minutes to subtract from the current time when creating the value for the certificate's `notBefore` attribute is 5 minutes (`notBeforeSkew=5`).

For details on individual parameters defined in the rule, see Table 3-13 on page 127. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

Certificate Extension Plug-in Modules

iPlanet Certificate Management Server (CMS) comes with a set of policy plug-in modules that enable you to add X.509 certificate extensions to certificates the server issues. This chapter explains those modules—it lists and briefly describes the modules that are installed with a Certificate Manager and Registration Manager, and then explains each one in detail.

The chapter has the following sections:

- Overview of Extension-Specific Policy Modules (page 132)
- AuthInfoAccessExt Plug-in Module (page 136)
- AuthorityKeyIdentifierExt Plug-in Module (page 144)
- BasicConstraintsExt Plug-in Module (page 147)
- CertificatePoliciesExt Plug-in Module (page 151)
- CertificateRenewalWindowExt Plug-in Module (page 156)
- CertificateScopeOfUseExt Plug-in Module (page 161)
- CRLDistributionPointsExt Plug-in Module (page 166)
- ExtendedKeyUsageExt Plug-in Module (page 171)
- GenericASN1Ext Plug-in Module (page 177)
- IssuerAltNameExt Plug-in Module (page 184)
- KeyUsageExt Plug-in Module (page 189)
- NameConstraintsExt Plug-in Module (page 202)
- NSCCommentExt Plug-in Module (page 211)
- NSCertTypeExt Plug-in Module (page 215)

- OCSPNoCheckExt Plug-in Module (page 220)
- PolicyConstraintsExt Plug-in Module (page 224)
- PolicyMappingsExt Plug-in Module (page 227)
- PrivateKeyUsagePeriodExt Plug-in Module (page 231)
- RemoveBasicConstraintsExt Plug-in Module (page 233)
- SubjectAltNameExt Plug-in Module (page 235)
- SubjectDirectoryAttributesExt Plug-in Module (page 241)
- SubjectKeyIdentifierExt Plug-in Module (page 245)

Overview of Extension-Specific Policy Modules

To enable you to add standard and private extensions to end-entity certificates, Certificate Management System provides a set of policy plug-in modules; each module enables you to add a particular extension to a certificate request. Plug-in modules are implemented as Java classes and are registered in the CMS policy framework. The Policy Plugin Registration tab of the CMS window (Figure 4-1) lists all the modules that are registered with a CMS instance.

Note that the name of the Java class for a policy plug-in module is in this format:

```
com.netscape.certsrv.policy.<plugin_name>
```

where <plugin_name> is the name of a plug-in module. For example, the Java class for the `AuthorityKeyIdentifierExt` module would be:

```
com.netscape.certsrv.policy.AuthorityKeyIdentifierExt
```

When deciding whether to add 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 every extension. For general guidelines on using extensions in certificates, see Appendix C, “Certificate and CRL Extensions.”

Figure 4-1 Extension policy modules registered with a Certificate Manager

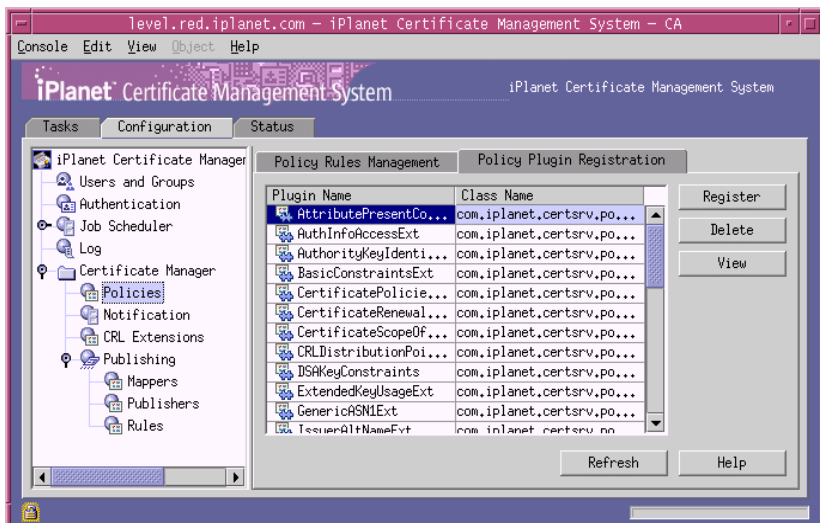


Table 4-1 lists extension-specific policy modules that are installed with a Certificate Manager. A Registration Manager installation also includes all the modules, except for the ones noted below:

- AuthorityKeyIdentifierExt
- BasicConstraintsExt
- NameConstraintsExt
- PolicyConstraintsExt
- PolicyMappingsExt

You can use these modules to configure a Certificate Manager and Registration Manager to add extensions to certificates. Both subsystems add extensions to a certificate request when it undergoes policy processing (see section “Policy Processor” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*). Keep in mind that the changes made to a request by a Registration Manager may be overwritten by a Certificate Manager when it subjects the request to its own policy checks.

Table 4-1 Default extension-specific policy plug-in modules

Plug-in module	Function
AuthInfoAccessExt	Adds the <i>Authority Information Access</i> extension to certificates. For details, see “AuthInfoAccessExt Plug-in Module” on page 136.
AuthorityKeyIdentifierExt	Adds the <i>Authority Key Identifier</i> extension to certificates. For details, see “AuthorityKeyIdentifierExt Plug-in Module” on page 144.
BasicConstraintsExt	Adds the <i>Basic Constraints</i> extension to certificates. For details, see “BasicConstraintsExt Plug-in Module” on page 147.
CertificatePoliciesExt	Adds the <i>Certificate Policies</i> extension to certificates. For details, see “CertificatePoliciesExt Plug-in Module” on page 151.
CertificateRenewalWindowExt	Adds the <i>Certificate Renewal Window</i> extension to certificates. For details, see “CertificateRenewalWindowExt Plug-in Module” on page 156.
CertificateScopeOfUseExt	Adds the <i>Certificate Scope of Use</i> extension to certificates. For details, see “CertificateScopeOfUseExt Plug-in Module” on page 161.
CRLDistributionPointsExt	Adds the <i>CRL Distribution Points</i> extension to certificates. For details, see “CRLDistributionPointsExt Plug-in Module” on page 166.
ExtendedKeyUsageExt	Adds the <i>Extended Key Usage</i> extension to certificates. For details, see “ExtendedKeyUsageExt Plug-in Module” on page 171.
GenericASN1Ext	Adds ASN.1 type custom extension to certificates. For details, see “GenericASN1Ext Plug-in Module” on page 177.
IssuerAltNameExt	Adds the <i>Issuer Alternative Name</i> extension to certificates. For details, see “IssuerAltNameExt Plug-in Module” on page 184.
KeyUsageExt	Adds the <i>Key Usage</i> extension to certificates. For details, see “KeyUsageExt Plug-in Module” on page 189.
NameConstraintsExt	Adds the <i>Name Constraints</i> extension to certificates. For details, see “NameConstraintsExt Plug-in Module” on page 202.
NSCCommentExt	Adds the <i>Netscape Certificate Comment</i> extension to certificates. For details, see “NSCCommentExt Plug-in Module” on page 211.
NSCertTypeExt	Adds the <i>Netscape Certificate Type</i> extension to certificates. For details, see “NSCertTypeExt Plug-in Module” on page 215.
OCSPNoCheckExt	Adds the <i>OCSP No Check</i> extension to certificates. For details, see “OCSPNoCheckExt Plug-in Module” on page 220.

Table 4-1 Default extension-specific policy plug-in modules (*Continued*)

Plug-in module	Function
PolicyConstraintsExt	Adds the <i>Policy Constraints</i> extension to certificates. For details, see “PolicyConstraintsExt Plug-in Module” on page 224.
PolicyMappingsExt	Adds the <i>Policy Mappings</i> extension to certificates. For details, see “PolicyMappingsExt Plug-in Module” on page 227.
PrivateKeyUsagePeriodExt	Adds the <i>Private Key Usage Period</i> extension to certificates. For details, see “PrivateKeyUsagePeriodExt Plug-in Module” on page 231.
RemoveBasicConstraintsExt	Detects and removes the <i>Basic Constraints</i> extension in certificate requests. For details, see “RemoveBasicConstraintsExt Plug-in Module” on page 233.
SubjectAltNameExt	Adds the <i>Subject Alternative Name</i> extension to certificates. For details, see “SubjectAltNameExt Plug-in Module” on page 235.
SubjectDirectoryAttributesExt	Adds a <i>Subject Directory Attributes</i> extension to certificates. For details, see “SubjectDirectoryAttributesExt Plug-in Module” on page 241.
SubjectKeyIdentifierExt	Adds the <i>Subject Key Identifier</i> extension to certificates. For details, see “SubjectKeyIdentifierExt Plug-in Module” on page 245.

As indicated in Table 4-1, Certificate Management System enables you to add almost all of the extensions defined in the PKIX standard RFC 2459 (<http://www.ietf.org/rfc/rfc2459.txt>). All modules have three features in common, enabling you to specify these:

- Whether to add the extension to certificates.
- The certificates to which the extension is to be added.
- Whether to mark the extension critical or noncritical.

By default, only noncritical extensions are added to certificates. This ensures that the resulting certificates can be used with all clients. If you add a critical extension, the resulting certificate can only be used by clients that support that extension.

Additionally, the server also provides a module for adding any custom, ASN.1 type extensions. If you determine that the default policy modules do not meet your requirements entirely, you can develop a custom module using CMS SDK. It is available in the form of Java Docs at this location:

```
<server_root>/cms_sdk/cms_jdk/javadocs
```

For general guidelines on developing custom policy modules and adding them to the CMS policy framework, take a look at the samples installed at these locations:

```
<server_root>/cms_sdk/cms_jdk/samples/policies
```

For instructions to configure a Certificate Manager and Registration Manager to use one or more of the policy modules, see section “Configuring Policy Rules for a Subsystem” in installChapter 18, “Setting Up Policies” of install*CMS Installation and Setup Guide*.

AuthInfoAccessExt Plug-in Module

The `AuthInfoAccessExt` plug-in module implements the authority information access extension policy. This policy enables you to configure Certificate Management System to add the *Authority Information Access Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) to certificates. The extension specifies how an application validating a certificate can access information, such as on-line validation services and CA policy statements, about the CA that has issued the certificate in which the extension appears. Note that this extension should not be used to point directly to the CRL location maintained by a CA; the CRL Distribution Points extension explained in “CRLDistributionPointsExt Plug-in Module” on page 166 allows you to reference to CRL locations.

The PKIX standard recommends that you may include the authority information access extension in end-entity and CA certificates and that the extension be marked noncritical. For general guidelines on setting the authority information access extension, see “authorityInfoAccess” on page 343.

The authority information access extension policy in Certificate Management System allows you to set the authority information access extension as defined in its X.509 definition. The policy enables you to specify any number of access points for CA information. For each access point, you can specify the access method, actual location that contains the additional information about the CA, and the mechanism for retrieving the information. The location can be specified in any of the following general-name forms: an rfc822name, a directory name, a DNS name, an EDI party name, a uniform resource indicator (URI), an IP address, an object identifier (OID), and any other name.

By default, the policy supports three access methods:

- `caIssuers` (this method is also identified by its OID, 1.3.6.1.5.5.7.48.2).

As specified in the PKIX standard, you should use the `caIssuers` method when the additional information is a list of parent CAs or CAs that have issued certificates superior to the CA that issued the certificate containing the extension. The certificate-using application may use the list of parent CAs referenced by the extension to determine the certification path and to check whether the path terminates at a point trusted by the certificate user.

When you use the `caIssuers` method, the access location referenced in the extension must take any of the following general-name forms:

- Uniform resource identifier (URI) if the information is available via HTTP, FTP, or LDAP.
 - An X.500 directory name if the information is available via the directory access protocol (DAP).
 - An `rfc822Name` if the information is available via electronic mail.
- `ocsp` (this method is also identified by its OID, 1.3.6.1.5.5.7.48.1).

The `ocsp` method indicates to the certificate-using client that it must use the OCSF protocol to access the location that contains additional information about the CA that has issued the certificate. You should use the `ocsp` method when you want to reference to the online validation authority that maintains the revocation status of certificates issued by the CA.

When you use the `ocsp` method, the access location referenced in the extension must be a uniform resource indicator (URI); this means, the location type must be `URL` and the location value must be the complete URL (including the port number) at which the online validation authority for the CA is listening for OCSF requests from OCSF-compliant clients.

- `renewal` (this method is also identified by its OID, 2.16.840.1.113730.16.1)

The `renewal` method works with the automated-certificate-renewal feature built into Netscape Personal Security Manager. When you use this method, the access location referenced in the extension must be a URI.

The built-in support for the `ocsp` access method and a URI value for the access location in the extension conform to the profile defined in RFC 2560 (see <http://www.ietf.org/rfc/rfc2560.txt>) for CAs that support the OCSF service. For details about OCSF support in Certificate Management System, see Chapter 21, “Setting Up an OCSF Responder” of *CMS Installation and Setup Guide*.

If you configure a Certificate Manager to publish CRLs to an OCSP responder and want to include the authority information access extension referencing to the responder, you should configure an instance of this policy as follows: access method is set to `ocsp`, name type is set to `URI`, and name value is set to the URL at which the OCSP responder listens to OCSP requests. This way, OCSP-compliant applications can verify the revocation status of certificates issued by the Certificate Manager by accessing the validation authority using the OCSP method.

During installation, Certificate Management System automatically creates an instance of the authority information access extension policy. See “AuthInfoAccessExt Rule” on page 143.

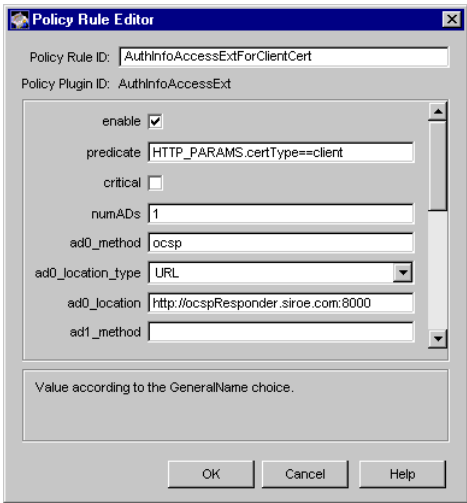
NOTE The CMS configuration file (`CMS.cfg`) includes a parameter named `jss.ocspcheck.enable`, which enables you to specify whether a CMS manager should use Online Certificate Status Protocol (OCSP) to verify the revocation status of the certificate it receives as a part of SSL client or server authentication (from clients or servers it makes connections with). If you change the value of this parameter to `true`, the CMS manager reads the Authority Information Access extension in the certificate and verifies the revocation status of the certificate from the OCSP responder specified in the extension.

Configuration Parameters of AuthInfoAccessExt

In the CMS configuration file, the `AuthInfoAccessExt` module is identified as `<subsystem>.Policy.impl.AuthInfoAccessExt.class=com.netscape.certsrv.policy.AuthInfoAccessExt`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `AuthInfoAccessExt`. Figure 4-2 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-2 Parameters defined in the AuthInfoAccessExt module



The configuration shown in Figure 4-2 creates a policy rule named `AuthInfoAccessExtForClientCert`, which enforces a rule that the server should add the authority information access extension to client certificates. The extension indicates that the online validation service (or the OSCSP responder) for the CA that has issued these certificates is at this URL:

`http://ocspResponder.siroe.com:8000`

The extension is marked noncritical (to comply with the PKIX recommendation).

Table 4-2 gives details about the configurable parameters defined in the `AuthInfoAccessExt` module.

Table 4-2 Description of parameters defined in the AuthInfoAccessExt module

Parameter	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server adds the authority information access extension to certificates specified by the <code>predicate</code> parameter.• If you disable the rule, the server does not add the extension to certificates; it ignores the values in the remaining fields.

Table 4-2 Description of parameters defined in the AuthInfoAccessExt module (*Continued*)

Parameter	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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: HTTP_PARAMS.certType==client</p>
critical	<p>Specifies whether the extension should be marked critical or noncritical in certificates specified by the predicate parameter. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).</p>
numADs	<p>Specifies the total number of access locations to be contained or allowed in the extension.</p> <p>By default, this field is set to 3 and the UI shows fields for configuring three locations. You can change the total number of locations by changing the value assigned to this parameter; there's no restriction on the total number of locations you can include in the extension.</p> <p>Note that each location has its own set of configuration parameters and you must specify appropriate values for each of those parameters; otherwise the policy rule will return an error. Each set of configuration parameters is distinguished by <n>, which is an integer derived from the value you assign in this field. For example, if you set the numADs parameter to 2, <n> would be 0 and 1.</p> <p>Permissible values: 0 or n.</p> <ul style="list-style-type: none"> • 0 specifies that no locations can be contained in the extension. • n specifies the total number of locations to be included in the extension; it must be an integer greater than zero. The default value is 3. <p>Example: 2</p>
ad<n>_method	<p>Specifies the access method for retrieving additional information about the CA that has issued the certificate in which the extension appears.</p> <p>Permissible values:</p> <ul style="list-style-type: none"> • ocsp (or 1.3.6.1.5.5.7.48.1). • caIssuers (or 1.3.6.1.5.5.7.48.2). • renewal (or 2.16.840.1.113730.16.1) <p>Example 1: ocsp</p> <p>Example 2: 1.3.6.1.5.5.7.48.1</p>

Table 4-2 Description of parameters defined in the AuthInfoAccessExt module *(Continued)*

Parameter	Description
<code>ad<n>_location_type</code>	<p>Specifies the general-name type for the location that contains additional information about the CA that has issued the certificate in which this extension appears.</p> <p>Permissible values: <code>rfc822Name</code>, <code>directoryName</code>, <code>dNSName</code>, <code>ediPartyName</code>, <code>URL</code>, <code>iPAddress</code>, <code>OID</code>, or <code>otherName</code>.</p> <ul style="list-style-type: none"> • Select <code>rfc822Name</code> if the location is an Internet mail address. • Select <code>directoryName</code> if the location is an X.500 directory name. • Select <code>dNSName</code> if the location is a DNS name. • Select <code>ediPartyName</code> if the location is a EDI party name. • Select <code>URL</code> if the location is a uniform resource identifier (default). • Select <code>iPAddress</code> if the location is an IP address. • Select <code>OID</code> if the location is an object identifier. • Select <code>otherName</code> if the location is in any other name form. <p>Example: <code>URL</code></p>
<code>ad<n>_location</code>	<p>Specifies the address or location to get additional information about the CA that has issued the certificate in which this extension appears.</p> <p>Permissible values: Depends on the location type you specified in the <code>ad<n>_location_type</code> field.</p> <ul style="list-style-type: none"> • If you selected <code>rfc822Name</code>, the value must be a valid Internet mail address in the <code>local-part@domain</code> format; see the definition of an <code>rfc822Name</code> as defined in RFC 822 (http://www.ietf.org/rfc/rfc0822.txt). You may use upper and lower case letters in the mail address; no significance is attached to the case. For example, <code>ocspResponder@siroe.com</code>. • If you selected <code>directoryName</code>, the value must be a string form of X.500 name, similar to the subject name in a certificate, in the RFC 2253 syntax (see http://www.ietf.org/rfc/rfc2253.txt). Note that RFC 2253 replaces RFC 1779. For example, <code>CN=corpDirectory, OU=IS, O=Siroe.com, C=US</code>. • If you selected <code>dNSName</code>, the value must be a valid domain name in the preferred-name syntax as specified in RFC 1034 (http://www.ietf.org/rfc/rfc1034.txt). You may use upper and lower case letters in the domain name; no significance is attached to the case. Do not use the string " " for the DNS name. Also don't use the DNS representation for Internet mail addresses; such identities should be encoded as <code>rfc822Name</code>. For example, <code>ocspResponder.siroe.com</code>. • If you selected <code>ediPartyName</code>, the value must be an IA5String. For example, <code>Siroe Corporation</code>.

Table 4-2 Description of parameters defined in the AuthInfoAccessExt module *(Continued)*

Parameter	Description
	<ul style="list-style-type: none"> If you selected <code>URL</code>, the value must be a non-relative universal resource identifier (URI) following the URL syntax and encoding rules specified in RFC 1738 (http://www.ietf.org/rfc/rfc1738.txt). That is, the name must include both a scheme (for example, <code>http</code>) and a fully qualified domain name or IP address of the host. For example, <code>http://ocspResponder.siroe.com:8000</code> If you selected <code>IPAddress</code>, the value must be a valid IP address specified in dot-separated numeric component notation. The syntax for specifying the IP address is as follows: For IP version 4 (IPv4), the address should be in the form specified in RFC 791 (http://www.ietf.org/rfc/rfc0791.txt). IPv4 address must be in the <code>n.n.n.n</code> format; for example, <code>128.21.39.40</code>. IPv4 address with netmask must be in the <code>n.n.n.n,m.m.m.m</code> format. For example, <code>128.21.39.40,255.255.255.00</code>. For IP version 6 (IPv6), the address should be in the form described in RFC 1884 (http://www.ietf.org/rfc/rfc1884.txt), with netmask separated by a comma. Examples of IPv6 addresses with no netmask are <code>0:0:0:0:0:0:13.1.68.3</code> and <code>FF01::43</code>. Examples of IPv6 addresses with netmask are <code>0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0</code> and <code>FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000</code>. If you selected <code>OID</code>, the value must be a unique, valid OID specified in dot-separated numeric component notation. Although you can invent your own OIDs for the purposes of evaluating and testing this server, in a production environment, you should comply with the ISO rules for defining OIDs and for registering subtrees of IDs. See Appendix B, "Object Identifiers" for information on allocating private OIDs. For example, <code>1.2.3.4.55.6.5.99</code>. If you selected <code>otherName</code>, the value must be the absolute path to the file containing the base-64 encoded string of the location. For example, <code>/opt/SUNWcertsrv/certsrv47/ext/aia/othername.txt</code>.

AuthInfoAccessExt Rule

The rule named `AuthInfoAccessExt` is an instance of the `AuthInfoAccessExt` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is disabled.
- The predicate expression (`predicate=HTTP_PARAMS.certType==client`) ensures that the policy is to be applied to client certificate requests processed by the server.
- The extension is marked noncritical (to comply with the PKIX recommendation).
- The total number of access locations to be contained or allowed in the extension is set to 1 (`numADs=1`).
- The access method for retrieving additional information about the CA that has issued the certificate in which the extension appears is set to OCSP (`ad0_method=ocsp`).
- The general-name type for the location that contains additional information about the CA that has issued the certificate in which the extension appears is set to URL (`ad0_location_type=URL`).
- The address or location to get additional information about the CA that has issued the certificate in which this extension appears is left blank for you to enter the URL at which the OCSP responder will service requests from OCSP-compliant clients.

Note that if you installed the Certificate Manager with its built-in OCSP service enabled, the policy rule will be enabled and the address location (`ad0_location=`) will be pointed to the Certificate Manager's nonSSL end-entity port. For example, if the nonSSL end-entity port of your Certificate Manager is 80, the URL would look like this: `http://ocspResponder.siroe.com:80/ocsp`

For details on individual parameters defined in the rule, see Table 4-2 on page 139. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section "Step 2. Modify Existing Policy Rules" in Chapter 18, "Setting Up Policies" of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section "Step 4. Add New Policy Rules" in the same chapter.

AuthorityKeyIdentifierExt Plug-in Module

The `AuthorityKeyIdentifierExt` plug-in module implements the authority key identifier extension policy. This policy enables you to configure Certificate Management System to add the *Authority Key Identifier Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) to certificates. The extension is used to identify the public key that corresponds to the private key used by a CA to sign certificates.

You should consider adding this extension to all certificates, especially CA certificates, issued by Certificate Management System. The reason is, in certain situations, a CA's public key may change (for example, when the key gets updated) or the CA may have multiple signing keys (either due to multiple concurrent key pairs or due to key changeover). In these cases, the CA ends up with more than one distinct key. When verifying a signature on a certificate, other applications need to know which key was used in the signature. The extension, if present in a certificate, enables applications (those that can use the extension) to identify the correct key to use in situations when multiple keys exist; the extension specifies the public key to be used to verify the signature on the certificate.

For general guidelines on setting the authority key identifier extension, see “`authorityKeyIdentifier`” on page 344.

The authority key identifier extension policy in Certificate Management System allows setting of the authority key identifier extension as defined in its X.509 definition with *key identifiers*. The policy enables you to specify what is to be done if the CA certificate does not have a subject key identifier extension—whether to use the a SHA-1 hash of the CA's subject public key information (carries the public key and identifies the algorithm with which the key is used) or skip adding the authority key identifier extension itself. For information on setting the subject key identifier extension in certificates, see “`SubjectKeyIdentifierExt Plug-in Module`” on page 245.

Note that PKIX and Federal PKI standards recommend against the use of `authorityCertIssuer` and `authorityCertSerialNumber` fields of the X.509 definition.

If enabled, the policy does the following:

- Sets the authority key identifier extension in certificates using the CA's key identifier in the CA's subject key identifier extension, if it exists. In the absence of a subject key identifier extension, the policy does either of the following (as specified by the configuration):

- Uses the SHA-1 hash of the CA's subject public key information as the key identifier. This option is compatible with Netscape Communicator when the CA does not have a subject public key identifier extension.
 - Does not set the authority key identifier extension.
- Adds a authority key identifier extension to an enrollment request if the extension does not already exist. If the extension exists in the request, for example from a CRMF request, the policy replaces the extension. In case of manual enrollments, after an agent approves the enrollment request, the policy accepts any authority key identifier extension that is already there.

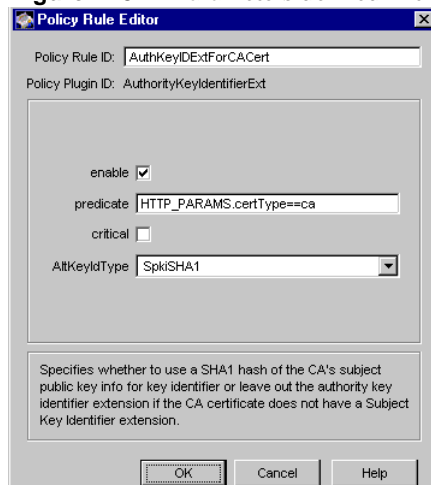
During installation, Certificate Management System automatically creates an instance of the authority key identifier extension policy. See “AuthorityKeyIdentifierExt Rule” on page 147.

Configuration Parameters of AuthorityKeyIdentifierExt

In the CMS configuration file, the AuthorityKeyIdentifierExt module is identified as `ca.Policy.impl.AuthorityKeyIdentifierExt.class=com.netscape.certsrv.policy.AuthorityKeyIdentifierExt`.

In the CMS window, the module is identified as AuthorityKeyIdentifierExt. Figure 4-3 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-3 Parameters defined in the AuthorityKeyIdentifierExt module



The configuration shown in Figure 4-3 creates a policy rule named `AuthKeyIDExtForCACert`, which enforces a rule that the server should set the authority key identifier extension in all CA certificates.

Table 4-3 gives details about each of these parameters.

Table 4-3 Description of parameters defined in the AuthorityKeyIdentifierExt module

Parameter	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none"> If you enable the rule and set the remaining parameters correctly, the server adds the authority key identifier extension to certificates specified by the <code>predicate</code> parameter. If you disable the rule, the server does not add the extension to certificates; it ignores the values in the remaining fields.
<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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==ca</code></p>
<code>critical</code>	<p>Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).</p>
<code>AltKeyIdType</code>	<p>Specifies what should be done if the CA certificate does not have a Subject Key Identifier extension.</p> <p>Permissible values: <code>SpkiSHA1</code> or <code>None</code>.</p> <ul style="list-style-type: none"> Select <code>SpkiSHA1</code> if you want the server to use a SHA-1 hash of the CA’s subject public key information (default). Select <code>None</code> if you don’t want the server to set the authority key identifier extension in certificates. <p>Example: <code>SpkiSHA1</code></p>

AuthorityKeyIdentifierExt Rule

The rule named `AuthorityKeyIdentifierExt` is an instance of the `AuthorityKeyIdentifierExt` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression is left blank so that the extension gets added to all certificates the server issues.
- The extension is marked noncritical (to comply with the PKIX recommendation).
- The rule specifies that a SHA-1 hash of the CA's subject public key info be used if the CA certificate does not have a Subject Key Identifier extension (`AltKeyIdType=SpkiSHA1`).

For details on individual parameters defined in the rule, see Table 4-3 on page 146. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

BasicConstraintsExt Plug-in Module

The `BasicConstraintsExt` plug-in module implements the basic constraints extension policy. This policy enables you to configure Certificate Management System to add the *Basic Constraints Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) in certificates. The 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.

You should consider adding this extension to all CA certificates (root as well as subordinate) issued by Certificate Management System. 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. For general guidelines on setting the basic constraints extension, see “basicConstraints” on page 345.

Because the basic constraints extension is a critical extension and is used by applications to determine the path length during certificate validation to chain up to the trusted CA, it's important that you set this extension correctly.

Also note that when a user submits a certificate request using the manual-enrollment method, the basic constraints extension is set on that request as per the configured policy, and then the request is queued for agent approval. When an agent approves the request, it is subjected to the configured policy again. If there's a change in the configuration of the basic constraints extension, the server may reject the agent-approved request. For the server to approve the request, the user will have to resubmit the request.

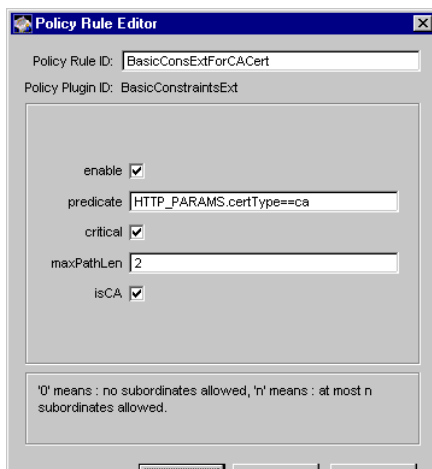
During installation, Certificate Management System automatically creates an instance of the basic constraints extension policy. See “BasicConstraintsExt Rule” on page 150.

Configuration Parameters of BasicConstraintsExt

In the CMS configuration file, the `BasicConstraintsExt` module is identified as `ca.Policy.impl.BasicConstraintsExt.class=com.netscape.certsrv.policy.BasicConstraintsExt`.

In the CMS window, the module is identified as `BasicConstraintsExt`. Figure 4-4 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-4 Parameters defined in the BasicConstraintsExt module



The configuration shown in Figure 4-4 creates a policy rule named `BasicConsExtForCACert`, which enforces a rule that the server should set the basic constraints extension in all CA certificates.

Table 4-4 gives details about each of these parameters.

Table 4-4 Description of parameters defined in the BasicConstraintsExt module

Parameter	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none"> • If you enable the rule and set the remaining parameters correctly, the server adds the basic constraints extension to certificates specified by the <code>predicate</code> parameter. • If you disable the rule, the server does not add the extension to certificates; it ignores the values in the remaining fields.
<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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==ca</code></p>
<code>critical</code>	<p>Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical (default). Uncheck the box if you want the server to mark the extension noncritical.</p>
<code>isCA</code>	<p>Specifies whether the certificate subject is a CA. If you select the option, the server checks the <code>maxPathLen</code> parameter and sets the specified path length in the certificate. If you deselect the option, the server treats the certificate subject as a non-CA and ignores the value specified for the <code>maxPathLen</code> parameter.</p>

Table 4-4 Description of parameters defined in the BasicConstraintsExt module *(Continued)*

Parameter	Description
maxPathLen	<p>Specifies the path length, the maximum number of CA certificates that may be chained below (subordinate to) the subordinate CA certificate being issued. Note that the path length you specify affects 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.</p> <p>The maxPathLen parameter has no effect if the extension is set in end-entity certificates.</p> <p>Permissible values: 0 or n. Make sure that the value you choose is less than the path length specified in the Basic Constraints extension of the CA signing certificate (owned by the CA that will issue these certificates).</p> <ul style="list-style-type: none">• 0 specifies that no subordinate CA certificates are allowed below the subordinate CA certificate being issued—that is, only an end-entity certificate may follow in the path.• n must be an integer greater than zero. It specifies at the most n subordinate CA certificates are allowed below the subordinate CA certificate being used.• If you leave the field blank, the path length defaults to a value that is determined by the path length set on the Basic Constraints extension in the issuer's certificate. If the issuer's path length is unlimited, the path length in the subordinate CA certificate will also be unlimited. If the issuer's path length is an integer greater than zero, the path length in the subordinate CA certificate will be set to a value that's one less than the issuer's path length; for example, if the issuer's path length is 4, the path length in the subordinate CA certificate will be set to 3. <p>Example: 2</p>

BasicConstraintsExt Rule

The rule named `BasicConstraintsExt` is an instance of the `BasicConstraintsExt` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression is set (`predicate=HTTP_PARAMS.certType==ca`) so that the extension gets added to CA certificates only.
- The extension is marked critical to comply with the PKIX recommendation.

- The path length field (`maxPathLen`) is left blank so that it defaults to a value that is determined by the path length set on the Basic Constraints extension in the issuer's certificate.

For details on individual parameters defined in the rule, see Table 4-4 on page 149. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

CertificatePoliciesExt Plug-in Module

The `CertificatePoliciesExt` plug-in module implements the certificate policies extension policy. This policy enables you to configure Certificate Management System to add the *Certificate Policies Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) in certificates. The extension contains a sequence of one or more policy statements, each indicating the policy under which the certificate has been issued and identifying the purposes for which the certificate may be used. Presence of this extension in certificates enables an application with specific policy requirements to compare its list of policies to the ones contained in a certificate during its validation; typically, such applications will have a list of policies (which they will accept) and compare the policies in the certificate to their list as a part validating the certificate.

To promote interoperability, the PKIX standard recommends that the policy statements or information terms should be included in certificates in the form of object identifiers (OIDs). For more information on OIDs, see Appendix B, “Object Identifiers.” This means, in order for the server to add this extension to any certificate it issues, you need to compose policy statements you want to include in the extension, define OIDs for these policy statements, and configure the server with these OIDs.

When determining whether to add this extension to certificates, keep in mind that if the extension exists in a certificate and if it is marked critical, the application validating the certificate must be able to interpret the extension (including the optional qualifiers, if any), or else it must reject the certificate. For general guidelines on setting the certificate policies extension, see “certificatePolicies” on page 346.

The certificate policies extension policy in Certificate Management System enables you to set the extension with the following information:

- The name of the your company or organization.
- The OID assigned to the policy statement you want to include in the certificate.
- A pointer (URI) to the published Certification Practice Statement (CPS).

Any company deploying its own PKI should make a CPS available to anyone who may come across certificates issued by the CA deployed in the PKI. (The reason for this is people outside the company intranet may receive signed email messages from an employee.) To see an example of a CPS, check this site:
<http://people.netscape.com/shadow/cps.html>

- A textual user notice (which the application validating the certificate can interpret and display).

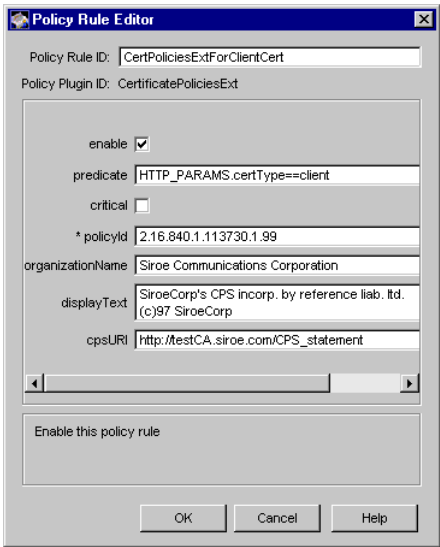
During installation, Certificate Management System automatically creates an instance of the certificate policies extension policy. See “CertificatePoliciesExt Rule” on page 155.

Configuration Parameters of CertificatePoliciesExt

In the CMS configuration file, the CertificatePoliciesExt module is identified as `<subsystem>.Policy.impl.CertificatePoliciesExt.class=com.netscape.certsrv.policy.CertificatePoliciesExt`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `CertificatePoliciesExt`. Figure 4-5 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-5 Parameters defined in the CertificatePoliciesExt module



The configuration shown in Figure 4-5 creates a policy rule named `CertPoliciesExtForClientCert`, which enforces a rule that the server should set the certificate policies extension in all client certificates.

Table 4-5 gives details about each of these parameters.

Table 4-5 Description of parameters defined in the CertificatePoliciesExt module

Parameter	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled.</p> <ul style="list-style-type: none">• Check the box to enable the rule (default). If you enable the rule and set the remaining parameters correctly, the server adds the certificate policies extension to certificates specified by the <code>predicate</code> parameter.• Uncheck the box to disable the rule. If you disable the rule, the server does not add the extension to certificates; it ignores the values in the remaining fields.
<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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client</code></p>

Table 4-5 Description of parameters defined in the CertificatePoliciesExt module *(Continued)*

Parameter	Description
<code>critical</code>	Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).
<code>policyId</code>	<p>Specifies the OID assigned to the policy statement you want to include in the extension. If you specify a valid OID, the server includes the OID in the extension.</p> <p>The <code>policyId</code>, if specified, identifies by number a particular textual statement prepared by your organization (which is specified by the parameter named <code>organizationName</code>, listed next in this table). For example, it might identify the organization as Siroe Corporation and notice number 1.2.3.4.5.6.99. Typically, applications validating the certificate will have a notice file containing the current set of notices for your company; these application will interpret the number in the certificate by extracting the notice text that corresponds to the number from the file and display it to the relying party.</p> <p>Permissible values: A unique, valid OID specified in dot-separated numeric component notation (see the example). Although you can invent your own OIDs for the purposes of evaluating and testing this server, in a production environment, you should comply with the ISO rules for defining OIDs and for registering subtrees of IDs. See Appendix B, “Object Identifiers” for information on allocating private OIDs.</p> <p>Example: 2.16.840.1.113730.1.99</p>
<code>organizationName</code>	<p>Specifies the name of the organization that owns the OID or is the owner of the policy statement referenced by the OID.</p> <p>Permissible values: The name of a company or its organizational unit.</p> <p>Example: Siroe Corporation</p>
<code>cpsURI</code>	<p>Specifies the location where the Certification Practice Statement published by the CA (that has issued the certificate) can be found.</p> <p>Permissible values: An IA5String value. The PKIX standard recommends that the pointer should be in the form of a URI.</p> <p>Example: <code>http://testCA.siroe.com/CPS_statement</code></p>

Table 4-5 Description of parameters defined in the CertificatePoliciesExt module *(Continued)*

Parameter	Description
displayText	<p>Specifies the textual statement to be included in certificates; this parameter corresponds to the <code>explicitText</code> field of the user notice. If you want to embed a textual statement (for example, your company’s legal notice) in certificates, then add that statement here. The text you enter here will be displayed to a relying party when the certificate is used or viewed.</p> <p>Note that certain applications may not have the capability to display this text. Also, embedding a policy statement in a certificate increases its size.</p> <p>If you specify values for both <code>policyId</code> and <code>displayText</code> parameters and if the application software cannot locate the notice text indicated by the <code>policyId</code> parameter, then it is supposed to display the embedded notice; otherwise, it’s supposed to display the information specified by the <code>policyId</code> parameter. (This feature is application specific and Certificate Management System has no control over it.)</p> <p>Permissible values: A string with up to 200 characters.</p> <p>Example: SiroeCorp’s CPS incorp. by reference liab. ltd. (c)97 SiroeCorp</p>

CertificatePoliciesExt Rule

The policy rule named `CertificatePoliciesExt` is an instance of the `CertificatePoliciesExt` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is disabled; for the rule to be effective, it must be enabled and configured appropriately.
- The predicate field is left blank so that the extension gets added to all certificates.
- The extension is marked noncritical (to comply with the PKIX recommendation).
- Other fields are left blank for you to enter the appropriate information.

For details on individual parameters defined in the rule, see Table 4-5 on page 153. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the

same chapter. For example, if you want to include different policy statements in different types of certificates, you should create multiple instances of the policy module and configure each instance with the appropriate policy OID and predicate expression.

CertificateRenewalWindowExt Plug-in Module

The `CertificateRenewalWindowExt` plug-in module implements the certificate renewal window extension policy. This policy enables you to configure Certificate Management System to add the *Certificate Renewal Window Extension* to certificates. The extension, which must be noncritical, aids in managing the life cycle of a certificate by specifying a process to follow for renewing a certificate and by defining a time window when automatic renewal of the certificate should be attempted.

Every certificate issued by Certificate Management System has a validity period beyond which it cannot be used. In order to continue to participate in the PKI-using system beyond this validity period, the entity owning the certificate must renew the certificate. Renewal of a certificate essentially means getting a new certificate for the existing key pair with a new validity time period (and updated attributes).

Once a certificate is issued, the owner of the certificate may attempt its renewal any time. To prevent certificate owners from renewing their certificates too often and thus reduce the overhead of processing new certificate requests, the CA can use a policy that restricts the time period when certificate renewal may occur. For example, the CA can use a policy that limits the renewal process to the last few weeks or days of validity of the certificate, thus defining a certificate renewal window. In general, the renewal window must be sufficient for the renewal to occur, but at the same time delay the renewal as long as possible to best utilize a certificate's life time.

The certificate-renewal process is often different than the enrollment process an entity uses to obtain the certificate; this is because the entity already owns a key pair that is associated with his or her identity. For example, in Certificate Management System, the certificate-renewal process for end users is different than the enrollment process they used to obtain the certificate. To renew their certificates, end users go to the certificate-renewal interface of Certificate Management System and submit their original certificates; for details, see section “Authentication of End Users During Certificate Renewal” in Chapter 15, “Setting Up End-User Authentication” of *CMS Installation and Setup Guide*.

Because the renewal process requires end users to remember when their certificates expire and renew them before the expiry date, some clients provide built-in support for automated renewal. Inclusion of the certificate renewal window extension in certificates is useful in a PKI setup with such clients; such a setup eliminates the need for the owner of the certificate to manually submit a renewal request to the CA and install the renewed certificate. For example, assume you have deployed clients that can automatically submit certificate-renewal requests to Certificate Management System. If you issue certificates with the certificate renewal window extension to these clients, they can then read this extension for the renewal window and automatically get the certificate renewed from the CA during that window.

For a PKI setup without clients that can handle automated certificate renewals, Certificate Management System enables administrators to easily manage certificate renewals using the following features:

- The renewal notification job, which reminds users to renew their certificates before they expire.
- The renewal constraints policy, which determines whether expired certificates can be renewed; see “RenewalConstraints Plug-in Module” on page 103.
- The renewal validity constraints policy, which controls when users can renew their certificates and what should be the validity period in renewed certificates; see “RenewalValidityConstraints Plug-in Module” on page 106.

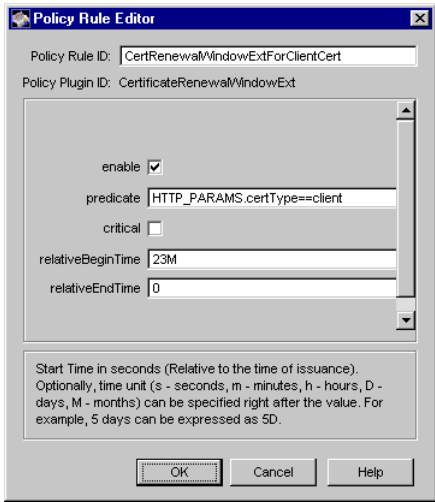
Unlike some of the other policy modules, Certificate Management System does not create an instance of the certificate renewal window extension policy during installation. If you want the server to add this extension to certificates, you must create an instance of the `CertificateScopeOfUseExt` module and configure it. For instructions, see section “Step 4. Add New Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*.

Configuration Parameters of CertificateRenewalWindowExt

In the CMS configuration file, the `CertificateRenewalWindowExt` module is identified as `<subsystem>.Policy.impl.CertificateRenewalWindowExt.class=com.netscape.certsrv.policy.CertificateRenewalWindowExt`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `CertificateRenewalWindowExt`. Figure 4-6 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-6 Parameters defined in the CertificateRenewalWindowExt module



The configuration shown in Figure 4-6 creates a policy rule named `CertRenewWindowExtForClientCert`, which enforces a rule that the server should set the certificate renewal window extension in client certificates only; the renewal window starts 30 days before a certificate expires and ends with certificate expiration.

Table 4-6 gives details about each of these parameters.

Table 4-6 Description of parameters defined in the CertificateRenewalWindowExt module

Parameter	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled.</p> <ul style="list-style-type: none">• Check the box to enable the rule (default). If you enable the rule and set the remaining parameters correctly, the server adds the certificate renewal window extension to certificates specified by the <code>predicate</code> parameter.• Uncheck the box to disable the rule. If you disable the rule, the server does not add the extension to certificates; it ignores the values in the remaining fields.
<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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client</code></p>

Table 4-6 Description of parameters defined in the CertificateRenewalWindowExt module (*Continued*)

Parameter	Description
<code>critical</code>	Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).
<code>relativeBeginTime</code>	<p>Specifies the first time automatic renewal of certificate that contains the extension should be attempted.</p> <p>Permissible values: 0 or n.</p> <ul style="list-style-type: none"> 0 specifies that the renewal window begins at the same time the certificate is issued; the <code>beginTime</code> field of the extension will be set to the time of certificate issuance. n specifies a future time for certificate renewal; the <code>beginTime</code> field of the extension will be set to the specified time since certificate issuance. You can specify the time period in seconds, minutes, hours, days, or months. Use the following suffixes to indicate the time unit. <ul style="list-style-type: none"> s - seconds m - minutes h - hours D - days M - months <p>For example, if you're issuing certificates with a validity period of two years and want the renewal window to begin a month before the certificates expire, and want to specify the interval in months, you would enter <code>23M</code> in this field. To specify the same validity interval in seconds, you would set the value to <code>59616000s</code> (23 months x 30 days x 24 hours x 60 minutes x 60 seconds).</p> <p>Example: <code>23M</code></p>

Table 4-6 Description of parameters defined in the CertificateRenewalWindowExt module *(Continued)*

Parameter	Description
relativeEndTime	<p>Specifies the last opportunity for automatic renewal of the certificate that contains this extension. Specifying a value for this parameter is optional; if you leave the field blank, the certificate-using application is expected to use the expiration date (notAfter value) in the certificate.</p> <p>Permissible values: 0 or n.</p> <ul style="list-style-type: none">0 specifies that the renewal window ends at the same time the certificate expires; the endTime field of the extension will be set to the time the certificate expires.n specifies a past or future time, in seconds, by which the certificate must be renewed; the endTime field of the extension will be set to the specified time since certificate issuance. You can specify the time period in seconds, minutes, hours, days, or months. Use the following suffixes to indicate the time unit. s - seconds m - minutes h - hours D - days M - months <p>For example, if you're issuing certificates with a validity period of two years and want the renewal window to end a month after the certificates expire, and want to specify the interval in months, you would enter 25M in this field. On the other hand, if you want the renewal window to end 15 days before certificates expire, then you would set the value to 705D ((23 months x 30 days) + 15 days).</p> <p>Note that if you choose to extend the renewal window after the expiration date of the certificate itself, your CA must maintain appropriate status information about the certificate during that window in order to allow appropriate authentication in the renewal process. (Automatic renewal may take place after the certificate has expired, when it is not valid for other purposes.)</p> <p>Example: 705D</p>

CertificateScopeOfUseExt Plug-in Module

The `CertificateScopeOfUseExt` plug-in module implements the certificate scope of use extension policy. This policy enables you to configure Certificate Management System to add the *Certificate Scope of Use Extension* to certificates. The extension enables you to specify a list of web sites that may request the use of a particular certificate for SSL client authentication, thus aiding certificate-using applications to select certificates to present to web sites and to control release of these certificates.

The SSL protocol provides a way for a client application to authenticate itself to a web site or server. SSL client authentication occurs upon request of the server, and proceeds by providing a certificate and a signature to the server. The client may have more than one certificate that could be used to perform this authentication. The SSL protocol provides a way for the server to indicate which certificate may be useful by listing issuing CAs in one of the SSL protocol messages.

By using a particular certificate for SSL client authentication, the client releases information about itself to the server. This information may include the name and key information contained in the certificate. It also releases the information that the client holds a certificate from a particular CA. This information may be of interest to the company running the server, for example to find users that have certificates from competing companies.

The certificate scope of use extension can be included in certificates to restrict the *scope-of-use* of the certificate for client authentication; the extension enables the certificate-using application to restrict the release of individual certificates to web sites requesting SSL client authentication.

The certificate scope of use extension policy in Certificate Management System enables you to include a list of name patterns that will match server DNS names where the certificate may be used. It's up to the certificate-using applications to use the values in this extension to filter the list of potential certificates to use for client authentication.

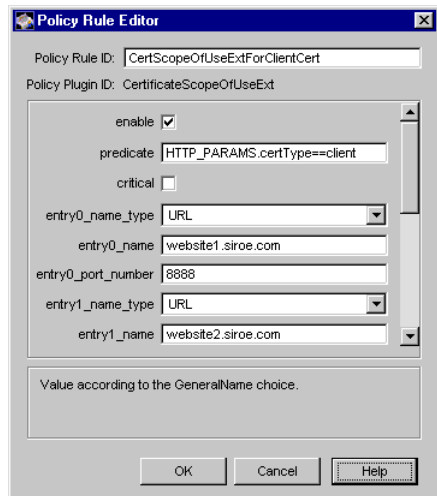
Unlike some of the other policy modules, Certificate Management System does not create an instance of the certificate scope of use extension policy during installation. If you want the server to add this extension to certificates, you must create an instance of the `CertificateScopeOfUseExt` module and configure it. For instructions, see section “Step 4. Add New Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*.

Configuration Parameters of CertificateScopeOfUseExt

In the CMS configuration file, the `CertificateScopeOfUseExt` module is identified as `<subsystem>.Policy.impl.CertificateScopeOfUseExt.class=com.netscape.certsrv.policy.CertificateScopeOfUseExt`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `CertificateScopeOfUseExt`. Figure 4-7 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-7 Parameters defined in the `CertificateScopeOfUseExt` module



The configuration shown in Figure 4-7 creates a policy rule named `CertScopeOfUseExtForClientCert`, which enforces a rule that the server should set the certificate scope of use extension in client certificates only.

Table 4-7 gives details about each of these parameters.

Table 4-7 Description of parameters defined in the CertificateScopeOfUseExt module

Parameter	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled.</p> <ul style="list-style-type: none"> • Check the box to enable the rule (default). If you enable the rule and set the remaining parameters correctly, the server adds the certificate scope of use extension to certificates specified by the <code>predicate</code> parameter. • Uncheck the box to disable the rule. If you disable the rule, the server does not add the extension to certificates; it ignores the values in the remaining fields.
<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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client</code></p>
<code>critical</code>	<p>Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).</p>
<code>numEntries</code>	<p>Specifies the total number of sites to be contained or allowed in the extension.</p> <p>By default, this field is set to 3 and the UI shows fields for configuring three sites. You can change the total number of sites by changing the value assigned to this parameter; there’s no restriction on the total number of sites you can include in the extension.</p> <p>Note that each site has its own set of configuration parameters and you must specify appropriate values for each of those parameters; otherwise the policy rule will return an error. Each set of configuration parameters is distinguished by <code><n></code>, which is an integer derived from the value you assign in this field. For example, if you set the <code>numEntries</code> parameter to 2, <code><n></code> would be 0 and 1.</p> <p>Permissible values: 0 or <code>n</code>.</p> <ul style="list-style-type: none"> • 0 specifies that no sites can be contained in the extension. • <code>n</code> specifies the total number of sites to be included in the extension; it must be an integer greater than zero. The default is 3. <p>Example: 2</p>

Table 4-7 Description of parameters defined in the CertificateScopeOfUseExt module *(Continued)*

Parameter	Description
entry<n>_name_type	<p>Specifies the general-name type for the site that you want to include in the extension.</p> <p>Permissible values: <code>rfc822Name</code>, <code>directoryName</code>, <code>dnsName</code>, <code>ediPartyName</code>, <code>URL</code>, <code>iPAddress</code>, <code>OID</code>, or <code>otherName</code>.</p> <ul style="list-style-type: none">• Select <code>rfc822Name</code> if the site is an Internet mail address.• Select <code>directoryName</code> if the site is an X.500 directory name.• Select <code>dnsName</code> if the site is a DNS name (default).• Select <code>ediPartyName</code> if the site is a EDI party name.• Select <code>URL</code> if the site is a uniform resource identifier.• Select <code>iPAddress</code> if the site is an IP address.• Select <code>OID</code> if the site is an object identifier.• Select <code>otherName</code> if the site is in any other name form. <p>Example: <code>URL</code></p>
entry<n>_name	<p>Specifies the general-name value for the site you want to include in the extension.</p> <p>Permissible values: Depends on the general-name type you selected in the <code>entry<n>_name_type</code> field.</p> <ul style="list-style-type: none">• If you selected <code>rfc822Name</code>, the value must be a valid Internet mail address in the <code>local-part@domain</code> format; see the definition of an <code>rfc822Name</code> as defined in RFC 822 (http://www.ietf.org/rfc/rfc0822.txt). You may use upper and lower case letters in the mail address; no significance is attached to the case. For example, <code>webSite@siroe.com</code>.• If you selected <code>directoryName</code>, the value must be a string form of X.500 name, similar to the subject name in a certificate, in the RFC 2253 syntax (see http://www.ietf.org/rfc/rfc2253.txt). Note that RFC 2253 replaces RFC 1779. For example, <code>CN=corpDirectory, OU=IS, O=Siroe.com, C=US</code>.• If you selected <code>dnsName</code>, the value must be a valid domain name in the preferred-name syntax as specified in RFC 1034 (http://www.ietf.org/rfc/rfc1034.txt). You may use upper and lower case letters in the domain name; no significance is attached to the case. Do not use the string “ ” for the DNS name. Also don’t use the DNS representation for Internet mail addresses; such identities should be encoded as <code>rfc822Name</code>. For example, <code>webSite.siroe.com</code>.

Table 4-7 Description of parameters defined in the CertificateScopeOfUseExt module *(Continued)*

Parameter	Description
	<ul style="list-style-type: none"> • If you selected <code>ediPartyName</code>, the value must be an IA5String. For example, <code>Siroe Corporation</code>. • If you selected <code>URL</code>, the value must be a non-relative URI, including both a scheme (for example, <code>http</code>) and a fully qualified domain name or IP address of the host. For example, <code>http://webSite.siroe.com</code>. • If you selected <code>iPAddress</code>, the value must be a valid IP address (IPv4 or IPv6) specified in dot-separated numeric component notation. The syntax for specifying the IP address is as follows: For IP version 4 (IPv4), the address should be in the form specified in RFC 791 (http://www.ietf.org/rfc/rfc0791.txt). IPv4 address must be in the <code>n.n.n.n</code> format; for example, <code>128.21.39.40</code>. IPv4 address with netmask must be in the <code>n.n.n.n,m.m.m.m</code> format. For example, <code>128.21.39.40,255.255.255.00</code>. For IP version 6 (IPv6), the address should be in the form described in RFC 1884 (http://www.ietf.org/rfc/rfc1884.txt), with netmask separated by a comma. Examples of IPv6 addresses with no netmask are <code>0:0:0:0:0:0:13.1.68.3</code> and <code>FF01::43</code>. Examples of IPv6 addresses with netmask are <code>0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0</code> and <code>FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000</code>. • If you selected <code>OID</code>, the value must be a unique, valid OID specified in dot-separated numeric component notation. Although you can invent your own OIDs for the purposes of evaluating and testing this server, in a production environment, you should comply with the ISO rules for defining OIDs and for registering subtrees of IDs. See Appendix B, “Object Identifiers” for information on allocating private OIDs. For example, <code>1.2.3.4.55.6.5.99</code>. • If you selected <code>otherName</code>, the value must be the absolute path to the file that contains the base-64 encoded string for the site. For example, <code>/opt/SUNWcertsrv/certsrv47//opt/SUNWcertsrv/certsrv47/ext/aia/othername.txt</code>.
<code>entry<n>_port_number</code>	<p>Specifies the port number.</p> <p>Example: 8888</p>

CRLDistributionPointsExt Plug-in Module

The `CRLDistributionPointsExt` plug-in module implements the CRL distribution points extension policy. This policy enables you to configure Certificate Management System to add the *CRL Distribution Points Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) to certificates. This extension, when present in a certificate, identifies one or more locations from where the application that is validating the certificate can obtain the CRL information (to verify the revocation status of the certificate).

For general guidelines on setting the CRL distribution points extension in certificates, see “`cRLDistributionPoints`” on page 347.

The CRL distribution points extension policy in Certificate Management System enables you to specify pointers to one or more CRL locations. The pointers can be in these forms: the name of the X.500 directory that stores the CRL, the URI to the location that contains the CRL, or both.

Note that in the current implementation, the policy supports only two name forms for distribution points, 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.

You can modify the policy to support any name form by making appropriate changes to the sample code provided for this purpose. The sample code is located here: `<server_root>/cms_sdk/cms_jdk/samples/policies`

During installation, Certificate Management System automatically creates an instance of the CRL distribution points extension policy. See “`CRLDistributionPointsExt Rule`” on page 170.

Configuration Parameters of CRLDistributionPointsExt

In the CMS configuration file, the `CRLDistributionPointsExt` module is identified as `<subsystem>.Policy.impl.CRLDistributionPointsExt.class=com.netscape.certsrv.policy.CRLDistributionPointsExt`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `CRLDistributionPointsExt`. Figure 4-8 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-8 Description of parameters defined in the `CRLDistributionPointsExt` module

The screenshot shows the 'Policy Rule Editor' window. At the top, 'Policy Rule ID' is 'CRLDistPtsExtForRouterCert' and 'Policy Plugin ID' is 'CRLDistributionPointsExt'. Below these are several configuration fields: 'enable' is checked, 'predicate' is 'HTTP_PARAMS.certType==CEP-Request', 'critical' is unchecked, 'numPoints' is '1', 'pointName0' is 'CN=testCA, OU=Research Dept, O=SiroeCorp, C=US', 'pointType0' is 'DirectoryName', 'reasons0' is empty, 'issuerName0' is empty, and 'issuerType0' is empty. A note at the bottom states: 'The name of the issuer that has signed the CRL maintained at this distribution point. The value depends on the issuer type.' At the bottom are 'OK', 'Cancel', and 'Help' buttons.

The configuration shown in Figure 4-8 creates a policy rule named `CRLDistPtsExtForRouterCert`, which enforces a rule that the server should set the CRL distribution point extension in router certificates; the CRL location is a X.500 directory.

Table 4-8 gives details about each of these parameters.

Table 4-8 Description of parameters defined in the `CRLDistributionPointsExt` module

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none"> If you enable the rule and set the remaining parameters correctly, the server adds the CRL distribution points extension to certificates specified by the <code>predicate</code> parameter. If you disable the rule, the server does not add the extension to certificates; it ignores the values in the remaining fields.

Table 4-8 Description of parameters defined in the CRLDistributionPointsExt module (*Continued*)

Parameter	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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==CEP-Request</code></p>
<code>critical</code>	<p>Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).</p>
<code>numPoints</code>	<p>Specifies the total number of CRL distribution points to be contained or allowed in the extension.</p> <p>By default, this field is set to 3 and the UI shows fields for configuring three distribution points. You can change the total number of distribution points by changing the value assigned to this parameter; there’s no restriction on the total number of distribution points you can include in the extension.</p> <p>Note that each distribution point has its own set of configuration parameters and you must specify appropriate values for each of those parameters; otherwise the policy rule will return an error. Each set of configuration parameters is distinguished by <code><n></code>, which is an integer derived from the value you assign in this field. For example, if you set the <code>numPoints</code> parameter to 2, <code><n></code> would be 0 and 1.</p> <p>Permissible values: 0 or <code>n</code>.</p> <ul style="list-style-type: none"> • 0 specifies that no distribution points can be contained in the extension. • <code>n</code> specifies the total number of distribution points to be included in the extension; it must be an integer greater than zero. The default is 3. <p>Example: 2</p>

Table 4-8 Description of parameters defined in the CRLDistributionPointsExt module (*Continued*)

Parameter	Description
pointName<n>	<p>Specifies the name of the CRL distribution point.</p> <p>Permissible values: Any supported name forms. By default, the name can be in any of the following formats:</p> <ul style="list-style-type: none"> • An X.500 directory name in the RFC 2253 syntax (see http://www.ietf.org/rfc/rfc2253.txt); note that RFC 2253 replaces RFC 1779. For example, the name would look similar to the subject name in a certificate, like this: CN=CA Central, OU=Research Dept, O=Siroe Corp, C=US • A URI; for example, it would look similar to this: http://testCA.siroe.com:80 • An RDN which specifies a location relative to the CRL Issuer. In this case, the value of the pointType attribute must be RelativeToIssuer.
pointType<n>	<p>Specifies the type of the CRL distribution point.</p> <p>Permissible values: DirectoryName, URI, or RelativeToIssuer. The type you select must correspond to the value in the pointName field.</p> <ul style="list-style-type: none"> • Select DirectoryName if the value in the pointName field is an X.500 directory name (default). • Select URI if the value in the pointName field is a uniform resource indicator. • Select RelativeToIssuer if the value in the pointName field is a location relative to the CRL Issuer. <p>Example: URI</p>
reasons<n>	<p>Specifies revocation reasons covered by the CRL maintained at the distribution point.</p> <p>Permissible values: A comma-separated list of the following constants.</p> <ul style="list-style-type: none"> • unused • keyCompromise • cACompromise • affiliationChanged • superseded • cessationOfOperation • certificateHold <p>Example: keyCompromise</p>

Table 4-8 Description of parameters defined in the CRLDistributionPointsExt module (*Continued*)

Parameter	Description
<code>issuerName<n></code>	<p>Specifies the name of the issuer that has signed the CRL maintained at distribution point.</p> <p>Permissible values: Any supported name forms. By default, the name can be in any of the following formats:</p> <ul style="list-style-type: none"> An X.500 directory name in the RFC 2253 syntax (see http://www.ietf.org/rfc/rfc2253.txt); note that RFC 2253 replaces RFC 1779. For example, the name would look similar to this: CN=CA Central, OU=Research Dept, O=Siroe Corp, C=US A URI; for example, it would look similar to this: http://testCA.siroe.com:80
<code>issuerType<n></code>	<p>Specifies the general-name type of the CRL issuer that has signed the CRL maintained at distribution point.</p> <p>Permissible values: <code>DirectoryName</code> or <code>URI</code>. The value you specify for this parameter must correspond to the value in the <code>issuerName</code> field.</p> <ul style="list-style-type: none"> Select <code>DirectoryName</code> if the value in the <code>issuerName</code> field is an X.500 directory name (default). Select <code>URI</code> if the value in the <code>issuerName</code> field is a uniform resource indicator. <p>Example: <code>DirectoryName</code></p>

CRLDistributionPointsExt Rule

The policy rule named `CRLDistributionPointsExt` is an instance of the `CRLDistributionPointsExt` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is disabled; for the rule to be effective, it must be enabled and configured appropriately.
- The predicate field is left blank so that the extension gets added to all certificates.
- The extension is marked noncritical (to comply with the PKIX recommendation).
- Other fields are left blank for you to enter the appropriate information.

For details on individual parameters defined in the rule, see Table 4-8 on page 167. It is important that you review this rule and make the appropriate changes required by your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter. For example, if you want to include different CRL distribution points in different types of certificates, you should create multiple instances of the policy module and configure each instance with the appropriate CRL distribution point and predicate expression.

ExtendedKeyUsageExt Plug-in Module

The `ExtendedKeyUsageExt` plug-in module implements the extended key usage extension policy. This policy enables you to configure Certificate Management System to add the *Extended Key Usage Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) to certificates. The extension identifies one or more purposes—in addition to or in place of the basic purposes indicated in the key usage extension—for which the certified public key may be used. For example, if the key usage extension identifies a key to be used for signing, the extended key usage extension can further narrow down the usage of the key for signing OCSP responses only or for signing Java applets only. (For information on key usage extension, see “KeyUsageExt Plug-in Module” on page 189.)

The PKIX standard suggests that organizations can define their own extended key usage purposes, if there’s a need. Each key purpose must be identified by an OID, which in turn must be defined in accordance with IANA or ITU-T Rec. X.660 | ISO/IEC/ITU 9834-1. The standard also recommends that the extension may be marked either critical or noncritical—mark the extension critical if you want to restrict the usage of the certificate only for one of the key-usage purposes indicated by the extension; mark the extension noncritical, when you want it to indicate the intended purposes of the key, and not restrict the use of the certificate to the indicated purposes (in this case, validating applications are expected to treat the extension as an advisory field and may use it to identify the key, not its usage purpose).

Table 4-9 lists the usages defined by PKIX for use with the extended key usage extension.

Table 4-9 PKIX usage definitions for the extended key usage extension

Usage	OID
Server authentication	1.3.6.1.5.5.7.3.1
Client authentication	1.3.6.1.5.5.7.3.2
Code signing	1.3.6.1.5.5.7.3.3
Email	1.3.6.1.5.5.7.3.4
IPSec end system	1.3.6.1.5.5.7.3.5
IPSec tunnel	1.3.6.1.5.5.7.3.6
IPSec user	1.3.6.1.5.5.7.3.7
Timestamping	1.3.6.1.5.5.7.3.8

Note that Windows 2000™ allows you to encrypt files on the hard disk, a feature known as encrypted file system (EFS), using certificates that contain the Extended Key Usage extension with the following two OIDs:

1.3.6.1.4.1.311.10.3.4 (this OID is for the EFS certificate)

1.3.6.1.4.1.311.10.3.4.1 (this OID is for the EFS recovery certificate)

The EFS recovery certificate is used by a recovery agent when a user loses the private key and the data encrypted with that key needs to be used. Certificate Management System supports the above two OIDs and allows you to issue certificates containing extended key usage extension with these OIDs.

Normal user certificates should be created with only the EFS OID, not the recovery OID.

For general guidelines on setting the extended key usage extension in certificates, see “extKeyUsage” on page 348.

The extended key usage extension policy in Certificate Management System allows setting of the key usage extension as defined in its X.509 definition. The policy enables you to specify OIDs, that identify key usages, in the extension.

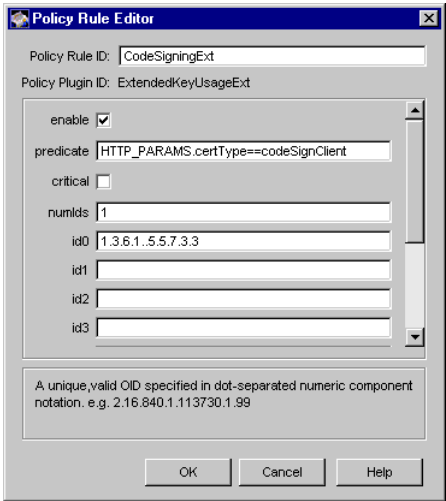
During installation, Certificate Management System automatically creates two instances of the extended key usage extension policy. See “CODESigningExt Rule” on page 175 and “OCSPSigningExt Rule” on page 176.

Configuration Parameters of ExtendedKeyUsageExt

In the CMS configuration file, the `ExtendedKeyUsageExt` module is identified as `<subsystem>.Policy.impl.ExtendedKeyUsageExt.class=com.netscape.certsrv.policy.ExtendedKeyUsageExt`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `ExtendedKeyUsageExt`. Figure 4-9 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-9 Parameters defined in the `ExtendedKeyUsageExt` module



The configuration shown in Figure 4-9 creates a policy rule named `CodeSigningExt`, which enforces a rule that the extended key usage extension should be set in object-signing certificates.

Table 4-10 gives details about each of these parameters.

Table 4-10 Description of parameters defined in the ExtendedKeyUsageExt module

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none"> If you enable the rule and set the remaining parameters correctly, the server adds the extended key usage extension to certificates specified by the <code>predicate</code> parameter. If you disable the rule, the server does not add the extension to certificates; it ignores the values in the remaining fields.
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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==codeSignClient</code></p>
critical	<p>Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical (default). Uncheck the box if you want the server to mark the extension noncritical.</p>
numIds	<p>Specifies the total number of key-usage purposes to be contained or allowed in the extension.</p> <p>By default, this field is set to 10 and the UI shows fields for configuring ten key-usage purposes. You can change the total number by changing the value assigned to this parameter; there's no restriction on the total number of key-usage purposes you can include in the extension.</p> <p>Note that for each key-usage purpose, you must specify a valid OID; otherwise the policy rule will return an error. Configuration parameters for each key-usage purposes is distinguished by <code><n></code>, which is an integer derived from the value you assign in this field. For example, if you set the <code>numIds</code> parameter to 2, <code><n></code> would be 0 and 1.</p> <p>Permissible values: 0 or <code>n</code>.</p> <ul style="list-style-type: none"> 0 specifies that no key-usage purposes can be contained in the extension. <code>n</code> specifies the total number of key-usage purposes to be included in the extension; it must be an integer greater than zero. The default value is 10. <p>Example: 1</p>

Table 4-10 Description of parameters defined in the ExtendedKeyUsageExt module *(Continued)*

Parameter	Description
<code>id<n></code>	<p>Specifies the OID that identifies a key-usage purpose.</p> <p>Permissible values: A unique, valid OID specified in the dot-separated numeric component notation. Depending on the key-usage purposes, you may choose to use the OIDs designated by PKIX (listed in Table 4-9 on page 172) or define your own OIDs. If you're defining your own OID, it should be in the registered subtree of IDs reserved for your company's use. Although you can invent your own OIDs for the purposes of evaluating and testing this server, in a production environment, you should comply with the ISO rules for defining OIDs and for registering subtrees of IDs. See Appendix B, "Object Identifiers" for information on allocating private OIDs.</p> <p>Example: 2.16.840.1.113730.1.99</p>

CODESigningExt Rule

The rule named `CODESigningExt` is an instance of the `ExtendedKeyUsageExt` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression is set (`HTTP_PARAMS.certType==codeSignClient`) so that the extension gets added to object signing certificates only—these certificates are used for signing objects.
- The extension is marked noncritical (to comply with the PKIX recommendation).
- The extension contains a single key-usage purpose, which is identified by an OID (`id0=1.3.6.1.5.5.7.3.3`). As shown in Table 4-9 on page 172, this OID is designated for code signing.

Note that this policy rule must remain enabled if you want Certificate Management System to issue object signing certificates with the correct extended key usage extension.

For details on individual parameters defined in the rule, see Table 4-10 on page 174. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section "Step 2. Modify Existing Policy Rules" in Chapter 18, "Setting Up Policies" of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section "Step 4. Add New Policy Rules" in the same chapter.

OCSPSigningExt Rule

The rule named `OCSPSigningExt` is an instance of the `ExtendedKeyUsageExt` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression is set (`HTTP_PARAMS.certType==ocspResponder`) so that the extension gets added to an OCSP responder certificate only—the certificate that corresponds to the key an online validation authority uses to sign OCSP responses.
- The extension is marked noncritical (to comply with the PKIX recommendation).
- The extension contains a single key-usage purpose, which is identified by an OID (`id0=1.3.6.1.5.5.7.3.9`).

Note that this policy rule must remain enabled if your PKI setup includes a CA-delegated OCSP responder and you want to issue an OCSP responder certificate to that server; the rule adds the extended key usage extension to an OCSP responder certificate indicating that the associated key can be used for signing OCSP responses.

Here's some background information that will help you understand why you should set this extension in OCSP responder certificates:

The online certificate status protocol (OCSP) enables OCSP-compliant applications to determine the revocation status of a certificate being validated. Certificate Management System supports the OCSP service—you can configure a Certificate Manager to publish CRLs to an online validation authority (also called OCSP responder); for details, see Chapter 21, “Setting Up an OCSP Responder” of *CMS Installation and Setup Guide*. If you configure Certificate Management System to work with an OCSP responder, OCSP-compliant applications in your PKI setup will be able to do real-time verification of certificates by querying the OCSP responder for their revocation status. Note that these applications will be able to query the OCSP responder only if the certificate being validated includes the authority information access extension indicating the location of the OCSP responder; for information on adding this extension to certificates, see “AuthInfoAccessExt Plug-in Module” on page 136.

When queried by an application on the status of a certificate, the OCSP responder sends a digitally signed response. To generate the signature, the responder needs to use a key. Because the signature needs to be verified by the application that sought the response, RFC 2560 recommends that the key used for signing an OCSP response must belong to one of the following:

- The CA that has issued the certificate, the revocation status of which is being requested.
- A trusted OCSP responder whose public key is trusted by the application that requested the revocation status of the certificate (as a part of validating the certificate).
- An OCSP responder that has been authorized by the CA (that has issued the certificate being validated) to sign OCSP responses for certificates issued by that CA.

In this type of deployment, the CA authorizes a responder to sign OCSP responses on its behalf by issuing a specially marked certificate to the responder. This certificate is called the *OCSP responder certificate*, and it enables OCSP-compliant applications to identify the responder as a *CA-designated responder*—a responder authorized to sign OCSP responses for all certificates issued by the CA. The special marking that the CA includes in the certificate is the extended key usage extension with a unique value, `OCSPSigning`. This extension value indicates to OCSP-compliant applications that the key associated with the certificate can be used for signing OCSP responses.

If you want to deploy a CA-delegated OCSP responder, the `OCSPSigningExt` rule enables you to add the extended key usage extension (with `OCSPSigning` value) to the OCSP responder certificate. In addition to this extension, the responder's signing certificate should also include the *OCSP no check* extension. For details, see “OCSPNoCheckExt Plug-in Module” on page 220.

GenericASN1Ext Plug-in Module

The `GenericASN1Ext` plug-in module implements the generic ASN.1 extension policy. This policy enables you to configure Certificate Management System to add custom extensions to certificates. Using this policy, you can add as many ASN.1 type based-extensions as required without having to write any code. Further, it eliminates the dependency on the command-line tools for generating base-64 encoded standard extensions from the x.509 extension classes.

The generic extension policy in Certificate Management System accepts custom extensions in the form of object identifiers (OIDs) and values as DER-encoded extension values. That is, for the server to add a custom extension to certificates it issues, you need to first define the extension and then configure the server with extension details.

Similar to a standard extension, you define a custom extension by defining an OID and a ASN.1 structure.

- The OID must be specified in the dot-separated numeric component notation (for example, 2.5.29.35). Although you can invent your own OIDs for the purposes of evaluating and testing the server, in a production environment, you should comply with the ISO rules for defining OIDs and for registering subtrees of IDs. See Appendix B, “Object Identifiers” for information on allocating private OIDs.
- The ASN.1 structure must be constructed from a sequence of DER-encoded extension values.

The resulting extension would look similar to the way a standard extension appears in certificates (as defined in RFC 2459):

```
Extension ::= SEQUENCE {
    extnID      OBJECT IDENTIFIER,
    critical    BOOLEAN DEFAULT FALSE,
    extnValue   OCTET STRING }
```

In the policy configuration, the `extnID` field is defined by the `oid` parameter, the `critical` field is defined by the `critical` parameter, and the `extnValue` field is defined by evaluating the expression in the `pattern` parameter, which in turn is defined by the `attribute` parameters. See Table 4-11 on page 180 for details on individual parameters.

Typically, the application receiving the certificate checks the extension ID to determine if it can recognize the ID. If it can, it uses the extension ID to determine the type of value used. When adding your custom extension to certificates, keep in mind that if the extension exists in a certificate and if it is marked critical, the application validating the certificate must be able to interpret the extension, or else it must reject the certificate. Since it's unlikely that all applications will be able to interpret your custom extensions, you should consider marking these extensions noncritical.

Note that each instance of the policy can be configured to add one custom extension only. To configure the server to add multiple custom extensions, create multiple instances of the module, each with a distinct name and appropriate configuration values. Also note that the policy allows you to encode simple (possibly nested) SEQUENCES. There is no support for CHOICE, SET, or ASN.1 tagging.

During installation, Certificate Management System automatically creates an instance of the generic ASN.1 extension policy. See “GenericASN1Ext Rule” on page 184.

Configuration Parameters of GenericASN1Ext

In the CMS configuration file, the `GenericASN1Ext` module is identified as `<subsystem>.Policy.impl.GenericASN1Ext.class=com.netscape.certsrv.policy.GenericASN1Ext`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `GenericASN1Ext`. Figure 4-10 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-10 Parameters defined in the GenericASN1Ext module

The screenshot shows the 'Policy Rule Editor' window for the 'GenericASN1Ext' module. The window contains the following configuration parameters:

- Policy Rule ID:** GenericASN1Ext
- Policy Plugin ID:** GenericASN1Ext
- enable:** ☒
- predicate:**
- critical:** ☐
- name:** testGenASN1Ext
- oid:** 2.4.5.99
- pattern:** {(0123(4{567})}
- attribute.0.type:** PrintableString
- attribute.0.source:** Value
- attribute.0.value:** 1st data in 1st sequence
- attribute.1.type:** IA5String
- attribute.1.source:** Value
- attribute.1.value:** 2nd data in 1st sequence
- attribute.2.type:** PrintableString
- attribute.2.source:** File
- attribute.2.value:** c:\temp\test.txt
- attribute.3.type:** OctetString
- attribute.3.source:** Value
- attribute.3.value:** 11:22:33:44 A0:B0:C0:D0:E0:F0

The configuration shown in Figure 4-10 defines a custom extension named testGenASN1Ext with OID 2.4.5.99. The extension is non-critical, and it will be added to all certificates issued by the server. The expected dumpasn1 output (see “dumpasn1 Tool” in CMS Command-Line Tools Guide) of the resulting extension, would look like this:

```
337 30 148: . . . . SEQUENCE {
340 06 3: . . . . . OBJECT IDENTIFIER '2 4 5 99'
345 04 140: . . . . . OCTET STRING, encapsulates {
348 30 137: . . . . . . SEQUENCE {
351 13 24: . . . . . . . PrintableString '1st data in 1st sequence'
377 16 24: . . . . . . . IA5String '2nd data in 1st sequence'
403 13 32: . . . . . . . PrintableString 'This is 3rd data in 1st
sequence'

437 04 10: . . . . . . . . OCTET STRING
: . . . . . . . . 11 22 33 44 A0 B0 C0 D0 E0 F0
449 30 37: . . . . . . . . SEQUENCE {
451 17 13: . . . . . . . . . UTCTime '000406070000Z'
466 30 8: . . . . . . . . . SEQUENCE {
468 01 1: . . . . . . . . . . BOOLEAN TRUE
471 06 3: . . . . . . . . . . OBJECT IDENTIFIER '2 4 5 100'
: . . . . . . . . . . }
476 04 10: . . . . . . . . . . OCTET STRING
: . . . . . . . . . . 11 22 33 44 A0 B0 C0 D0 E0 F0
: . . . . . . . . . . }
: . . . . . . . . . . }
: . . . . . . . . . . }
: . . . . . . . . . . }
```

Table 4-11 describes the configurable parameters of the generic ASN.1 extension policy module.

Table 4-11 Description of parameters defined in the GenericASN1Ext module

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule. Uncheck the box to disable the rule (default).</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server adds the configured extension to certificates specified by the predicate parameter.• If you disable the rule, the server does not add the extension to certificates; it ignores the values in the remaining fields.

Table 4-11 Description of parameters defined in the GenericASN1Ext module (*Continued*)

Parameter	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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client</code></p>
critical	<p>Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).</p> <p>In general, you should make custom extensions noncritical if you want your certificates supported by other applications. (Other applications most likely will not understand your extension.)</p>
name	<p>Specifies the name of the extension. The name is displayed when users view the details of a certificate that includes the extension.</p> <p>Permissible values: A unique name that corresponds to the OID specified by the <code>oid</code> parameter.</p> <p>Example: <code>myCorp'sExtension</code></p>
oid	<p>Specifies the OID assigned to the extension.</p> <p>Permissible values: A valid OID specified in dot-separated numeric component notation (see the example). Although you can invent your own OIDs for the purposes of evaluating and testing this server, in a production environment, you should comply with the ISO rules for defining OIDs and for registering subtrees of IDs. See Appendix B, “Object Identifiers” for information on allocating private OIDs.</p> <p>Example: <code>1.22.333.444.55.666</code></p>

Table 4-11 Description of parameters defined in the GenericASN1Ext module *(Continued)*

Parameter	Description
pattern	<p>Specifies the pattern of the extension.</p> <p>Permissible values: The pattern can be any sequence of supported ASN.1 type. Rules for formulating the pattern are as follows:</p> <ul style="list-style-type: none">• Each data component in the pattern must be represented by it's predefined attribute identifier, 0 to 9, and each sequence must be grouped by a pair of curly brackets, {}.• Each attribute identifier represented in the pattern must be fully defined in the extension. For example, if you want to include attribute identifier 0, you must specify values for <code>attribure.0.type</code>, <code>attribure.0.source</code>, and <code>attribure.0.value</code> parameters. <p>No default value is assigned to this parameter.</p> <p>Example: {{012}34}</p>
attribute.<n>.type	<p>Specifies the data type for attribute n, where n is an identifier assigned to identify parameters pertaining to a specific attribute. The value of n can be 0 to 9.</p> <p>Permissible values: Integer, IA5String, OctetString, PrintableString, UTCTime, OID, or Boolean.</p> <ul style="list-style-type: none">• Select Integer for extensions that have ASN.1 INTEGER values (default). It's case insensitive and accepts an integer in decimal notation as value.• Select IA5String for extensions that have ASN.1 IA5String values. It's case insensitive and accepts any normal string as value.• Select OctetString for extensions that have ASN.1 OCTET STRING values. It's case insensitive and the value is dependent on data source. If the data source is Value, the value must be in colon-separated, ASCII hexadecimal encoding notation. If the data source is File, the server reads the attribute value from the file specified.• Select VisualString for extensions that have printing character sets of International ASCII.• Select PrintableString for extensions that have ASN.1 PrintableString values. It's case insensitive and accepts any normal string as value.• Select UTCTime for site-defined extensions that have ASN.1 UTCTime values.• Select OID for extensions that have ASN.1 OBJECT IDENTIFIER values.• Select Boolean for extensions that have ASN.1 BOOLEAN values. It's case insensitive and accepts true or false as value. <p>Example: Integer</p>

Table 4-11 Description of parameters defined in the GenericASN1Ext module (*Continued*)

Parameter	Description
<code>attribute.<n>. source</code>	<p>Specifies the data source for attribute <i>n</i> in the extension, where <i>n</i> is an identifier assigned to identify parameters pertaining to a specific attribute. The value of <i>n</i> can be 0 to 9.</p> <p>In some cases, it may be preferable to put the value of an attribute in a file, instead of specifying it in the configuration parameters. This may be the case if the value of the attribute is a long text file or octet-string, for example.</p> <p>Permissible values: <i>Value</i> or <i>File</i>. (The attribute's <i>value</i> parameter is interpreted according to the value specified for this parameter.)</p> <ul style="list-style-type: none"> <i>Value</i> specifies that the attribute's <i>value</i> parameter is literally the value to be inserted in the extension (default). <i>File</i> specifies that the attribute's <i>value</i> parameter is a fully-qualified pathname of a file containing the value to be inserted in the extension. <p>Example: <i>Value</i></p>
<code>attribute.<n>. value</code>	<p>Specifies the data value for attribute <i>n</i>, where <i>n</i> is an identifier assigned to identify parameters pertaining to a specific attribute. The value of <i>n</i> can be 0 to 9.</p> <p>Permissible values: Depends on the data type and source you selected.</p> <ul style="list-style-type: none"> If the data type is <i>Integer</i>, enter an integer in decimal notation as value. For example, 1234567890. If the data type is <i>IA5String</i>, enter a normal string as value. For example, <i>Test of IA5String</i>. If the data type is <i>OctetString</i> and if the data source is <i>Value</i>, enter the value in colon-separated ASCII hexadecimal encoding notation. For example, 11:22:33:44:A0:B0:C0:D0:E0:F0. If data source is <i>File</i>, enter the complete file path, including the filename, in the specified format. When specifying file path in a Window NT system do not use the NT native file separator, the backward slash (\). Use Unix style file separator, the forward slash (/), instead. For example, C:/customExt/octet_string_value.txt. If the data type is <i>PrintableString</i>, enter a normal string as value. For example, <i>This_is_a_printable_string</i>. If the data type is <i>UTCTime</i>, enter a date in mm/dd/yy format. For example, April 5, 2000 would be 4/5/00 and October 10, 2001 would be 10/10/01. If the data type is <i>OID</i>, enter a valid OID. For example, 11.33.234.99. If the data type is <i>Boolean</i>, enter <i>true</i> or <i>false</i> as value. For example, <i>true</i>.

GenericASN1Ext Rule

The rule named `GenericASN1Ext` is an instance of the `GenericASN1Ext` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is disabled; for the rule to be effective, it must be enabled and configured appropriately.
- The predicate field is left blank so that the extension gets added to all certificates the server issues.
- The extension is marked noncritical (to comply with the PKIX recommendation).
- Other fields are left blank for you to enter the appropriate information.

For details on individual parameters defined in the rule, see Table 4-11 on page 180. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

IssuerAltNameExt Plug-in Module

The `IssuerAltNameExt` plug-in module implements the issuer alternative name extension policy. This policy enables you to configure Certificate Management System to add the *Issuer Alternative Name Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) to certificates. This extension enables binding of or associating Internet style identities—such as Internet electronic mail address, a DNS name, an IP address, and a uniform resource indicator (URI)— with the certificate issuer.

For general guidelines on setting the issuer alternative name extension, see “`issuerAltName`” on page 350.

The issuer alternative name extension policy in Certificate Management System allows setting of the issuer alternative name extension as defined in its X.509 definition. The policy enables you to associate the following alternative identities to a CA, by including them in the extension:

- An `rfc822` name
- A directory name

- A DNS name
- An EDI party name
- A uniform resource indicator (URI)
- An IP address
- An object identifier (OID)
- Other Name

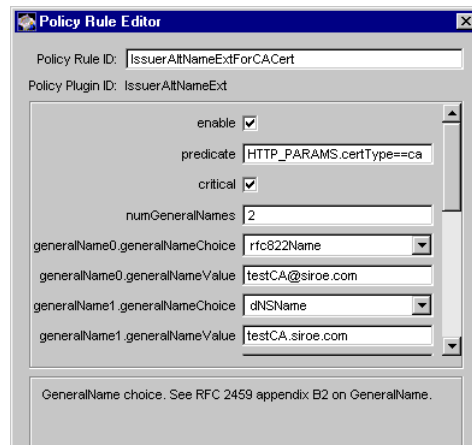
Unlike some of the other policy modules, Certificate Management System does not create an instance of the issuer alternative name extension policy during installation. If you want the server to add this extension to certificates, you must create an instance of the `IssuerAltNameExt` module and configure it. For instructions, see section “Step 4. Add New Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*.

Configuration Parameters of IssuerAltNameExt

In the CMS configuration file, the `IssuerAltNameExt` module is identified as `<subsystem>.Policy.impl.IssuerAltNameExt.class=com.netscape.certsrv.policy.IssuerAltNameExt`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `IssuerAltNameExt`. Figure 4-11 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-11 Parameters defined in the IssuerAltNameExt module



The configuration shown in Figure 4-11 creates a policy rule named `IssuerAltNameExtForCACert`, which enforces a rule that the server should set the issuer alternative name extension in CA certificates only.

Table 4-12 gives details about each of these parameters.

Table 4-12 Description of parameters defined in the `IssuerAltNameExt` module

Parameter	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server adds the issuer alternative name extension to all certificates specified by the <code>predicate</code> parameter.• If you disable the rule, the server doesn't add the extension to certificates; it ignores the values in the remaining fields.
<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 (default). To form a predicate expression, see section "Using Predicates in Policy Rules" in Chapter 18, "Setting Up Policies" of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==ca</code></p>
<code>critical</code>	<p>Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical (default). Uncheck the box if you want the server to mark the extension noncritical.</p>
<code>numGeneralNames</code>	<p>Specifies the total number of alternative names or identities permitted in the extension. Note that each name has a set of configuration parameters—<code>generalName<n>.generalNameChoice</code> and <code>generalName<n>.generalNameValue</code>—and you must specify appropriate values for each of those parameters; otherwise the policy rule will return an error. You can change the total number of identities by changing the value specified in this field; there's no restriction on the total number of identities you can include in the extension. Each set of configuration parameters is distinguished by <code><n></code>, which is an integer derived from the value you assign in this field. For example, if you set the <code>numGeneralNames</code> parameter to 2, <code><n></code> would be 0 and 1.</p> <p>Permissible values: 0 or <code>n</code>.</p> <ul style="list-style-type: none">• 0 specifies that no identities can be contained in the extension (default).• <code>n</code> specifies the total number of identities to be included in the extension; it must be an integer greater than zero. The default value is 8. <p>Example: 2</p>

Table 4-12 Description of parameters defined in the IssuerAltNameExt module (*Continued*)

Parameter	Description
<code>generalName<n>.generalNameChoice</code>	<p>Specifies the general-name type for the alternative name you want to include in the extension.</p> <p>Permissible values: <code>rfc822Name</code>, <code>directoryName</code>, <code>dNSName</code>, <code>ediPartyName</code>, <code>URL</code>, <code>iPAddress</code>, <code>OID</code>, or <code>otherName</code>.</p> <ul style="list-style-type: none"> • Select <code>rfc822Name</code> if the alternative name is an Internet mail address (default). • Select <code>directoryName</code> if the alternative name is an X.500 directory name. • Select <code>dNSName</code> if the alternative name is a DNS name. • Select <code>ediPartyName</code> if the alternative name is a EDI party name. • Select <code>URL</code> if the alternative name is a uniform resource locator (URL). • Select <code>iPAddress</code> if the alternative name is an IP address. • Select <code>OID</code> if the alternative name is an object identifier. • Select <code>otherName</code> if the alternative name is in any other name form. <p>Example: <code>rfc822Name</code></p>
<code>generalName<n>.generalNameValue</code>	<p>Specifies the general-name value for the alternative name you want to include in the extension.</p> <p>Permissible values: Depends on the general-name type you selected in the <code>generalName<n>.generalNameChoice</code> field.</p> <ul style="list-style-type: none"> • If you selected <code>rfc822Name</code>, the value must be a valid Internet mail address in the <code>local-part@domain</code> format; see the definition of an <code>rfc822Name</code> as defined in RFC 822 (http://www.ietf.org/rfc/rfc0822.txt). You may use upper and lower case letters in the mail address; no significance is attached to the case. For example, <code>testCA@siroe.com</code>. • If you selected <code>directoryName</code>, the value must be a string form of X.500 name, similar to the subject name in a certificate, in the RFC 2253 syntax (see http://www.ietf.org/rfc/rfc2253.txt). Note that RFC 2253 replaces RFC 1779. For example, <code>CN=CA Corp,OU=Research Dept,O=Siroe Corp,C=US</code>.

Table 4-12 Description of parameters defined in the IssuerAltNameExt module (*Continued*)

Parameter	Description
	<ul style="list-style-type: none"> If you selected <code>dNSName</code>, the value must be a valid domain name in the preferred-name syntax as specified by RFC 1034 (http://www.ietf.org/rfc/rfc1034.txt). You may use upper and lower case letters in the domain name; no significance is attached to the case. Do not use the string “ ” for the DNS name. Also don't use the DNS representation for Internet mail addresses; such identities should be encoded as <code>rfc822Name</code>. For example, <code>testCA.siroe.com</code>. If you selected <code>ediPartyName</code>, the value must be an <code>IA5String</code>. For example, <code>Siroe Corporation</code>. If you selected <code>URL</code>, the value must be a non-relative universal resource identifier (URI) following the URL syntax and encoding rules specified in RFC 1738. That is, the name must include both a scheme (for example, <code>http</code>) and a fully qualified domain name or IP address of the host. For example, <code>http://testCA.siroe.com</code>. If you selected <code>ipAddress</code>, the value must be a valid IP address (IPv4 or IPv6) specified in dot-separated numeric component notation. The syntax for specifying the IP address is as follows: For IP version 4 (IPv4), the address should be in the form specified in RFC 791 (http://www.ietf.org/rfc/rfc0791.txt). IPv4 address must be in the <code>n.n.n.n</code> format; for example, <code>128.21.39.40</code>. IPv4 address with netmask must be in the <code>n.n.n.n,m.m.m.m</code> format. For example, <code>128.21.39.40,255.255.255.00</code>. For IP version 6 (IPv6), the address should be in the form described in RFC 1884 (http://www.ietf.org/rfc/rfc1884.txt), with netmask separated by a comma. Examples of IPv6 addresses with no netmask are <code>0:0:0:0:0:0:13.1.68.3</code> and <code>FF01::43</code>. Examples of IPv6 addresses with netmask are <code>0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0</code> and <code>FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000</code>. If you selected <code>OID</code>, the value must be a unique, valid OID specified in the dot-separated numeric component notation. Although you can invent your own OIDs for the purposes of evaluating and testing this server, in a production environment, you should comply with the ISO rules for defining OIDs and for registering subtrees of IDs. See Appendix B, “Object Identifiers” for information on allocating private OIDs. For example, <code>1.2.3.4.55.6.5.99</code>. If you selected <code>otherName</code>, the value must be the absolute path to the file that contains the base-64 encoded string of the alternative name. For example, <code>/opt/SUNWcertsrv/certsrv47/ext/ian/othername.txt</code>.

KeyUsageExt Plug-in Module

The `KeyUsageExt` plug-in module implements the key usage extension policy. This policy enables you to configure Certificate Management System to add the *Key Usage Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) to certificates. The extension specifies the purposes for which the key contained in a certificate should be used—for example, it specifies whether the key should be used for data signing, key encipherment, or data encipherment—and thus enables you to restrict the usage of a key pair to predetermined purposes.

The key usage extension is a string of boolean bit-flags, each bit identifying the purpose for which a key is to be used. Table 4-13 lists the bits and their designated purposes.

Table 4-13 Key usage extension bits and designated purposes

Bit	Purpose
0	digitalSignature
1	nonRepudiation
2	keyEncipherment
3	dataEncipherment
4	keyAgreement
5	keyCertSign
6	cRLSign
7	encipherOnly
8	decipherOnly

You can restrict the purposes for which a key pair (and thus the corresponding certificate) should be used by setting the appropriate key-usage bits. For example, if you want to restrict a key pair to be used for digital signature only, when issuing the certificate you would add the key usage extension to the certificate with `digital_signature` bit (or bit 0) set. For general guidelines on setting the key usage extension in certificates, see “keyUsage” on page 351.

Note that you can specify which bits in the extension are to be set on both server and client sides:

- On the server side, you set the bits by modifying the appropriate configuration parameters that are defined in the key usage extension policy.
- On the client side, bits set in the key usage extension are formed from pre-defined HTTP input variables that can be embedded as hidden values in the enrollment forms. You specify which bits are to be set by adding the appropriate HTTP variables to the enrollment forms. Table 4-14 lists the HTTP input variables that correspond to key usage extension bits.

NOTE For all certificates, the key-usage-bits set on the server side (which is governed by the policy) override the ones set on the client side.

Table 4-14 HTTP input variables for key usage extension bits

HTTP input variable	Key usage extension bit
digital_signature	digitalSignature (bit 0)
non_repudiation	nonRepudiation (bit 1)
key_encipherment	keyEncipherment (bit 2)
data_encipherment	dataEncipherment (bit3)
key_agreement	keyAgreement (bit4)
key_certsign	keyCertsign (bit5)
crl_sign	cRLSign (bit6)
encipher_only	encipherOnly (bit7)
decipher_only	decipherOnly (bit8)

During installation, Certificate Management System automatically creates multiple instances of the key usage extension policy suitable for various types of certificates that you may want the server to issue. The default instances are named as follows:

- CMCertKeyUsageExt (For details, see “CMCertKeyUsageExt Rule” on page 196.)
- RMCertKeyUsageExt (For details, see “RMCertKeyUsageExt Rule” on page 197.)
- ServerCertKeyUsageExt (For details, see “ServerCertKeyUsageExt Rule” on page 198.)

- ClientCertKeyUsageExt (For details, see “ClientCertKeyUsageExt Rule” on page 199.)
- ObjSignCertKeyUsageExt (For details, see “ObjSignCertKeyUsageExt Rule” on page 201.)
- CRLSignCertKeyUsageExt (For details, see “CRLSignCertKeyUsageExt” on page 202.)

It is important that you review each policy instance and make the appropriate changes required by your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

Additionally, as you’ll notice in Figure 4-13 through Figure 4-17, the default enrollment forms provided for requesting various types of certificates (see “Enrollment Forms” on page 57) include the appropriate HTTP input variables that correspond to the key-usage bits. By default only variables that correspond to key-usage bits that need to be set are included in the form.

Typically, you won’t have to change the key-usage bit setting by editing the enrollment forms as you can do this easily by making the appropriate changes to the policy instance (bits set on the server side override the ones set on the client side). However, if you want to add new variables on the client side, you can do that too. Be sure to add the new variable in the following format:

```
<input type="HIDDEN" name="variable_name" value=true>
```

where, `variable_name` can be any of the HTTP input variables listed in Table 4-14.

The value of an HTTP input variable corresponding to a key-usage bit must be either `true` or `false`; any other value is considered equivalent to `false`. For example, a value `true` would be interpreted as `false` by the server. Note that values `true` and `false` are case insensitive.

Configuration Parameters of KeyUsageExt

In the CMS configuration file, the `KeyUsageExt` module is identified as `<subsystem>.Policy.impl.KeyUsageExt.class=com.netscape.certsrv.policy.KeyUsageExt`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `KeyUsageExt`. Figure 4-12 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-12 Parameters defined in the KeyUsageExt module

The screenshot shows the 'Policy Rule Editor' window. At the top, the 'Policy Rule ID' is 'KeyUsageExtForClientCert' and the 'Policy Plugin ID' is 'KeyUsageExt'. Below these, there are several configuration options:

- enable**: checked (checkbox)
- predicate**: HTTP_PARAMS.certType==client (text field)
- critical**: checked (checkbox)
- digitalSignature**: true (dropdown menu)
- nonRepudiation**: true (dropdown menu)
- keyEncipherment**: true (dropdown menu)
- dataEncipherment**: false (dropdown menu)
- keyAgreement**: false (dropdown menu)
- keyCertSign**: false (dropdown menu)
- crlSign**: false (dropdown menu)
- encipherOnly**: false (dropdown menu)
- decipherOnly**: false (dropdown menu)

At the bottom, there is a note: 'true means always set this bit, false means don't set this bit, HTTP_INPUT means get this bit from the HTTP input'. Below the note are three buttons: 'OK', 'Cancel', and 'Help'.

The configuration shown in Figure 4-12 creates a policy rule named `KeyUsageExtForClientCert`, which enforces a rule that the server should set the key usage extension (`digitalSignature`, `nonRepudiation`, and `keyEncipherment` bits) in client certificates.

Table 4-15 gives details about each of these parameters.

Table 4-15 Description of parameters defined in the KeyUsageExt module

Parameter	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none"> • If you enable the rule, the server checks the key usage extension bits specified in the remaining fields, and adds the extension with those bits to certificates specified by the <code>predicate</code> parameter. • If you disable the rule, the server does not add the extension to certificates; it ignores the key usage extension-specific bits specified in the policy configuration and in the enrollment forms.
<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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client</code></p>
<code>critical</code>	<p>Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical (default). Uncheck the box if you want the server to mark the extension noncritical.</p>
<code>digitalSignature</code>	<p>Specifies whether to set the <code>digitalSignature</code> bit (or bit 0) of the key usage extension in certificates specified by the <code>predicate</code> parameter.</p> <p>Permissible values: <code>true</code>, <code>false</code>, or <code>HTTP_INPUT</code>.</p> <ul style="list-style-type: none"> • Select <code>true</code> if you want the server to set the bit (default). • Select <code>false</code> if you don’t want the server to set the bit. • Select <code>HTTP_INPUT</code> if you want the server to check the certificate request for the HTTP input variable corresponding to the <code>digitalSignature</code> bit and set the bit accordingly. If the variable is set to <code>true</code>, the server sets the bit. If the variable doesn’t exist or if it is set to <code>false</code> (or any other value), the server doesn’t set the bit.

Table 4-15 Description of parameters defined in the KeyUsageExt module *(Continued)*

Parameter	Description
<code>nonRepudiation</code>	<p>Specifies whether to set the <code>nonRepudiation</code> bit (or bit 1) of the key usage extension in certificates specified by the <code>predicate</code> parameter.</p> <p>Permissible values: <code>true</code>, <code>false</code>, or <code>HTTP_INPUT</code>.</p> <ul style="list-style-type: none"> • Select <code>true</code> if you want the server to set the bit (default). • Select <code>false</code> if you don't want the server to set the bit. • Select <code>HTTP_INPUT</code> if you want the server to check the certificate request for the HTTP input variable corresponding to the <code>nonRepudiation</code> bit and set the bit accordingly. If the variable is set to <code>true</code>, the server sets the bit. If the variable doesn't exist or if it is set to <code>false</code> (or any other value), the server doesn't set the bit.
<code>keyEncipherment</code>	<p>Specifies whether to set the <code>keyEncipherment</code> bit (or bit 2) of the key usage extension in certificates specified by the <code>predicate</code> parameter.</p> <p>Permissible values: <code>true</code>, <code>false</code>, or <code>HTTP_INPUT</code>.</p> <ul style="list-style-type: none"> • Select <code>true</code> if you want the server to set the bit (default). • Select <code>false</code> if you don't want the server to set the bit. • Select <code>HTTP_INPUT</code> if you want the server to check the certificate request for the HTTP input variable corresponding to the <code>keyEncipherment</code> bit and set the bit accordingly. If the variable is set to <code>true</code>, the server sets the bit. If the variable doesn't exist or if it is set to <code>false</code> (or any other value), the server doesn't set the bit.
<code>dataEncipherment</code>	<p>Specifies whether to set the <code>dataEncipherment</code> bit (or bit 3) of the key usage extension in certificates specified by the <code>predicate</code> parameter.</p> <p>Permissible values: <code>true</code>, <code>false</code>, or <code>HTTP_INPUT</code>.</p> <ul style="list-style-type: none"> • Select <code>true</code> if you want the server to set the bit (default). • Select <code>false</code> if you don't want the server to set the bit. • Select <code>HTTP_INPUT</code> if you want the server to check the certificate request for the HTTP input variable corresponding to the <code>dataEncipherment</code> bit and set the bit accordingly. If the variable is set to <code>true</code>, the server sets the bit. If the variable doesn't exist or if it is set to <code>false</code> (or any other value), the server doesn't set the bit.

Table 4-15 Description of parameters defined in the KeyUsageExt module *(Continued)*

Parameter	Description
keyAgreement	<p>Specifies whether to set the <code>keyAgreement</code> bit (or bit 4) of the key usage extension in certificates specified by the <code>predicate</code> parameter.</p> <p>Permissible values: <code>true</code>, <code>false</code>, or <code>HTTP_INPUT</code>.</p> <ul style="list-style-type: none"> • Select <code>true</code> if you want the server to set the bit (default). • Select <code>false</code> if you don't want the server to set the bit. • Select <code>HTTP_INPUT</code> if you want the server to check the certificate request for the HTTP input variable corresponding to the <code>keyAgreement</code> bit and set the bit accordingly. If the variable is set to <code>true</code>, the server sets the bit. If the variable doesn't exist or if it is set to <code>false</code> (or any other value), the server doesn't set the bit.
keyCertSign	<p>Specifies whether to set the <code>keyCertSign</code> bit (or bit 5) of the key usage extension in certificates specified by the <code>predicate</code> parameter.</p> <p>Permissible values: <code>true</code>, <code>false</code>, or <code>HTTP_INPUT</code>.</p> <ul style="list-style-type: none"> • Select <code>true</code> if you want the server to set the bit (default). • Select <code>false</code> if you don't want the server to set the bit. • Select <code>HTTP_INPUT</code> if you want the server to check the certificate request for the HTTP input variable corresponding to the <code>keyCertSign</code> bit and set the bit accordingly. If the variable is set to <code>true</code>, the server sets the bit. If the variable doesn't exist or if it is set to <code>false</code> (or any other value), the server doesn't set the bit.
cRLSign	<p>Specifies whether to set the <code>cRLSign</code> bit (or bit 6) of the key usage extension in certificates specified by the <code>predicate</code> parameter.</p> <p>Permissible values: <code>true</code>, <code>false</code>, or <code>HTTP_INPUT</code>.</p> <ul style="list-style-type: none"> • Select <code>true</code> if you want the server to set the bit (default). • Select <code>false</code> if you don't want the server to set the bit. • Select <code>HTTP_INPUT</code> if you want the server to check the certificate request for the HTTP input variable corresponding to the <code>cRLSign</code> bit and set the bit accordingly. If the variable is set to <code>true</code>, the server sets the bit. If the variable doesn't exist or if it is set to <code>false</code> (or any other value), the server doesn't set the bit.

Table 4-15 Description of parameters defined in the KeyUsageExt module *(Continued)*

Parameter	Description
encipherOnly	<p>Specifies whether to set the encipherOnly bit (or bit 7) of the key usage extension in certificates specified by the predicate parameter.</p> <p>Permissible values: true, false, or HTTP_INPUT.</p> <ul style="list-style-type: none"> • Select true if you want the server to set the bit (default). • Select false if you don't want the server to set the bit. • Select HTTP_INPUT if you want the server to check the certificate request for the HTTP input variable corresponding to the encipherOnly bit and set the bit accordingly. If the variable is set to true, the server sets the bit. If the variable doesn't exist or if it is set to false (or any other value), the server doesn't set the bit.
decipherOnly	<p>Specifies whether to set the decipherOnly bit (or bit 8) of the key usage extension in certificates specified by the predicate parameter.</p> <p>Permissible values: true, false, or HTTP_INPUT.</p> <ul style="list-style-type: none"> • Select true if you want the server to set the bit (default). • Select false if you don't want the server to set the bit. • Select HTTP_INPUT if you want the server to check the certificate request for the HTTP input variable corresponding to the decipherOnly bit and set the bit accordingly. If the variable is set to true, the server sets the bit. If the variable doesn't exist or if it is set to false (or any other value), the server doesn't set the bit.

CMCertKeyUsageExt Rule

The policy rule named `CMCertKeyUsageExt` is an instance of the `KeyUsageExt` module. This rule is for setting the appropriate key-usage bits in Certificate Manager CA signing certificates; see section “CA Signing Key Pair and Certificate” in Chapter 14, “Managing CMS Keys and Certificates” of *CMS Installation and Setup Guide*. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression (`predicate=HTTP_PARAMS.certType==ca`) ensures that the rule is applied only to CA signing certificate requests.
- The extension is marked noncritical (to comply with the PKIX recommendation).

- The server is configured to set `digitalSignature`, `nonRepudiation`, `keyCertSign`, and `cRLSign` bits in CA signing certificates. Notice that the key-usage bits specified in the default policy rule match the bits specified in the enrollment form (`ManCAEnroll.html`) for requesting CA signing certificates (see Figure 4-13).

Figure 4-13 Key usage bit-specific variables in the Certificate Manager enrollment form



RM CertKeyUsageExt Rule

The policy rule named `RM CertKeyUsageExt` is an instance of the `KeyUsageExt` module. This rule is for setting the appropriate key-usage bits in Registration Managers' signing certificates; see section "Signing Key Pair and Certificate" in Chapter 14, "Managing CMS Keys and Certificates" of *CMS Installation and Setup Guide*. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression (`HTTP_PARAMS.certType==ra`) ensures that the rule is applied only to Registration Manager signing certificate requests.
- The extension is marked noncritical (to comply with the PKIX recommendation).

- The server is configured to set `digitalSignature` and `nonRepudiation` bits in Registration Manager signing certificates. Notice that the key-usage bits specified in the default policy rule match the bits specified in the enrollment form (`ManRAEnroll.html`) for requesting Registration Manager signing certificates (see Figure 4-14).

Figure 4-14 Key usage bit-specific variables in the Registration Manager enrollment form



ServerCertKeyUsageExt Rule

The policy rule named `ServerCertKeyUsageExt` is an instance of the `KeyUsageExt` module. This rule is for setting the appropriate key-usage bits in SSL server certificates. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression (`HTTP_PARAMS.certType==server`) ensures that the rule is applied only to SSL server certificate requests.
- The extension is marked noncritical (to comply with the PKIX recommendation).
- The server is configured to set `digitalSignature`, `nonRepudiation`, `keyEncipherment`, and `dataEncipherment` bits in SSL server certificates. Notice that the key-usage bits specified in the default policy rule match the bits specified in the enrollment form (`ManServerEnroll.html`) for requesting SSL server certificates (see Figure 4-15).

Figure 4-15 Key usage bit-specific variables in the SSL server certificate enrollment form


```

VIM - /export/cms/bin/cert/web/ee/ManServerEnroll.html
Window Edit Options Help

</tr>
<tr>
  <td valign="TOP" colspan="2">
    <table border="0" width="100%" cellspacing="0" cellpadding="6" bgcolor="#cccccc"
background="art/gray90.gif">
      <tr>
        <td>
          <div align="RIGHT">
            <input type="submit" value="Submit" name="submit" width="72">
              <input type="hidden" name="requestFormat" value="pkcs10">
              <input type="hidden" name="certType" value="server">
              <!-- for Netscape Certificate Type Extension -->
              <input type="HIDDEN" value="true" name="ssl_server">
              <!-- for Key Usage Extension -->
              <input type="HIDDEN" name="digital_signature" value=true>
              <input type="HIDDEN" name="non_repudiation" value=true>
              <input type="HIDDEN" name="key_encipherment" value=true>
              <input type="HIDDEN" name="data_encipherment" value=true>
            
            <input type="reset" value="Reset" name="reset" width="72">
            
            <input type="button" value="Help"
onclick="help('/manual/ee_guide/htmlvt.htm#Server

```

ClientCertKeyUsageExt Rule

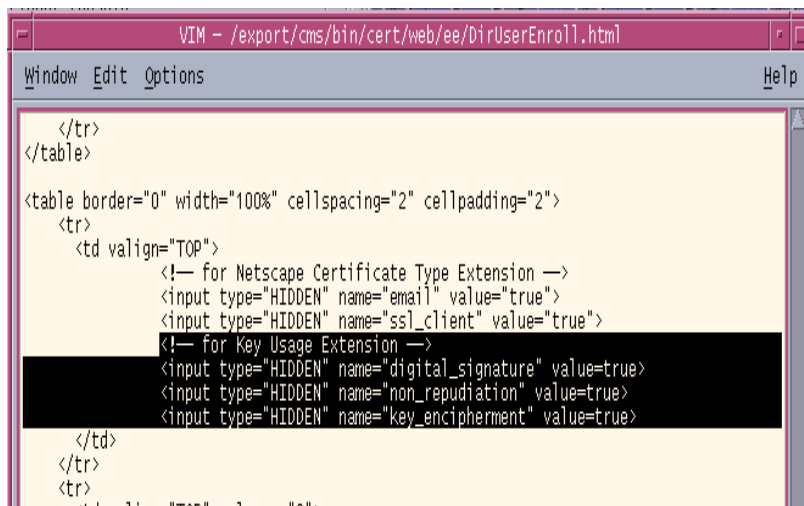
The policy rule named `ClientCertKeyUsageExt` is an instance of the `KeyUsageExt` module. This rule is for setting the appropriate key-usage bits in SSL client certificates. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression (`HTTP_PARAMS.certType==client`) ensures that the rule is applied only to SSL client certificate requests.
- The extension is marked noncritical (to comply with the PKIX recommendation).
- The server is configured to set `digitalSignature`, `nonRepudiation`, and `keyEncipherment` key-usage bits in SSL client certificates.

Notice that the key-usage bits specified in the default policy rule match the bits specified in the enrollment form for requesting SSL client certificates. Figure 4-16 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 in Table 4-14 on page 190. Note that by default three key-usage bits—`digitalSignature`, `nonRepudiation`, and `keyEncipherment`—are enabled and the remaining bits are disabled.

Additionally, also notice the HTTP variables for the Netscape certificate type extension: the values indicate that the certificate is meant for S/MIME and SSL client authentication use only. (For details on Netscape certificate type extension, see “NSCertTypeExt Plug-in Module” on page 215.)

Figure 4-16 Key usage extension bits in the directory-based enrollment form



Keep in mind that for requesting client certificates, there are many enrollment forms. You may be using a combination of them:

- Certificate-based enrollment forms (CertBasedDualEnroll.html, CertBasedEncryptionEnroll.html, or CertBasedSingleEnroll.html)
- Directory-based enrollment form (DirUserEnroll.html)
- Directory- and PIN-based enrollment form (DirPinUserEnroll.html)
- Manual enrollment form (ManUserEnroll.html)
- NIS-based enrollment form (NISEnroll.html)
- Portal enrollment form (PortalEnrollment.html)

For details about these forms, see “Enrollment Forms” on page 57.

Each of these forms embed HTTP input variables (for key-usage bits) that are considered appropriate for the certificate being requested using that form. If you want, you may create additional instances of the key usage extension policy, one each for each client certificate enrollment form and configure these instances as appropriate. Be sure to use the correct predicate expression to distinguish the certificates to thus avoid setting incorrect bits.

ObjSignCertKeyUsageExt Rule

The policy rule named `ObjSignCertKeyUsageExt` is an instance of the `KeyUsageExt` module. This rule is for setting the appropriate key-usage bits in object signing certificates. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression `(predicate=HTTP_PARAMS.certType==objSignClient)` ensures that the rule is applied to only object signing certificate requests.
- The extension is marked noncritical (to comply with the PKIX recommendation).
- The server is configured to set `digitalSignature` and `keyCertSign` bits in object-signing certificates. Notice that the key-usage bits specified in the default policy rule match the bits specified in the enrollment form (`ManObjSignEnroll.html`) for requesting object-signing certificates (see Figure 4-17).

Figure 4-17 Key usage extension bits in the object signing certificate enrollment form



CRLSignCertKeyUsageExt

The policy rule named `CrlSignCertKeyUsageExt` is an instance of the `KeyUsageExt` module. This rule is for setting the appropriate key-usage bits in a CRL signing certificate. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression `(predicate=HTTP_PARAMS.certType==caCrlSigning)` ensures that the rule is applied to only CRL signing certificate requests.
- The server is configured to set the `cRLSign` bit in CRL signing certificates.

NameConstraintsExt Plug-in Module

The `NameConstraintsExt` plug-in module implements the name constraints extension policy. This policy enables you to configure Certificate Management System to add the *Name Constraints Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) to certificates. The extension is used in CA certificates to indicate a name space within which subject names or subject alternative names in subsequent certificates in a certification path or chain should be located.

Various standards describe how the name constraints extension should be processed during certificate verification. It's beyond the scope of this document to explain this. For general guidelines on setting the name constraints extension in certificates, see “nameConstraints” on page 354.

The policy implemented in Certificate Management System allows setting of the name constraints extension in any form as defined in its X.509 definition; the policy enables you to specify the number of subtrees permitted and excluded in the extension. It is up to applications to process the extension as described in the standards.

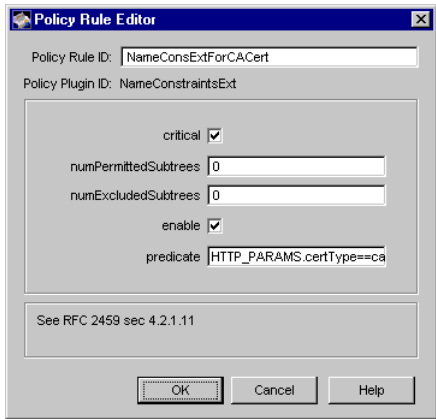
During installation, Certificate Management System automatically creates an instance of the name constraints extension policy. See “NameConstraintsExt Rule” on page 210.

Configuration Parameters of NameConstraintsExt

In the CMS configuration file, the NameConstraintsExt module is identified as `ca.Policy.impl.NameConstraintsExt.class=com.netscape.certsrv.policy.NameConstraintsExt`.

In the CMS window, the module is identified as NameConstraintsExt. Figure 4-18 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-18 Parameters defined in the NameConstraintsExt module



The configuration shown in Figure 4-18 creates a policy rule named NameConsExtForCACert, which enforces a rule that the server should set the name constraints extension as a critical extension in CA certificates.

Table 4-16 gives details about each of these parameters.

Table 4-16 Description of parameters defined in the NameConstraintsExt module

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server adds the name constraints extension to all certificates specified by the <code>predicate</code> parameter.• If you disable the rule, the server doesn't add the extension to certificates; it ignores the values in the remaining fields.
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 (default). To form a predicate expression, see section "Using Predicates in Policy Rules" in Chapter 18, "Setting Up Policies" of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==ca</code></p>
critical	<p>Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical (default). Uncheck the box if you want the server to mark the extension noncritical.</p>
numPermittedSubtrees	<p>Specifies the total number of subtrees to be permitted in the extension. Note that each permitted subtree has a set of configuration parameters and you must specify appropriate values for each of these parameters; otherwise the policy rule will return an error.</p> <p>You can change the total number of permitted subtrees by changing the value in this field; there's no restriction on the total number of permitted subtrees you can include in the extension. Each set of configuration parameters is distinguished by <code><n></code>, which is an integer derived from the value you assign in this field. For example, if you set the <code>numPermittedSubtrees</code> parameter to 2, <code><n></code> would be 0 and 1.</p> <p>Permissible values: 0 or <code>n</code>.</p> <ul style="list-style-type: none">• 0 specifies that no permitted subtrees can be contained in the extension.• <code>n</code> specifies the total number of permitted subtrees to be included in the extension; it must be an integer greater than zero. The default value is 8. <p>Example: 2</p>

Table 4-16 Description of parameters defined in the NameConstraintsExt module *(Continued)*

Parameter	Description
numExcludedSubtrees	<p>Specifies the total number of subtrees to be excluded in the extension. Note that each excluded subtree has a set of configuration parameters and you must specify appropriate values for each of these parameters; otherwise the policy rule will return an error.</p> <p>You can change the total number of excluded subtrees by changing the value in this field; there's no restriction on the total number of excluded subtrees you can include in the extension. Each set of configuration parameters is distinguished by <n>, which is an integer derived from the value you assign in this field. For example, if you set the numExcludedSubtrees parameter to 2, <n> would be 0 and 1.</p> <p>Permissible values: 0 or n.</p> <ul style="list-style-type: none">• 0 specifies that no excluded subtrees can be contained in the extension.• n specifies the total number of excluded subtrees to be included in the extension; it must be an integer greater than zero. The default value is 8. <p>Example: 2</p>
permittedSubtrees<n>. base.generalNameChoice	<p>Specifies the general-name type for the permitted subtree you want to include in the extension.</p> <p>Permissible values: rfc822Name, directoryName, dNSName, ediPartyName, URI, iPAddress, registeredID, or otherName.</p> <ul style="list-style-type: none">• Select rfc822Name if the subtree is an Internet mail address (default).• Select directoryName if the subtree is an X.500 directory name.• Select dNSName if the subtree is a DNS name.• Select ediPartyName if the subtree is a EDI party name.• Select URL if the subtree is a uniform resource locator.• Select iPAddress if the subtree is an IP address.• Select OID if the subtree is an object identifier.• Select otherName if the subtree is in any other name form. <p>Example: directoryName</p>

Table 4-16 Description of parameters defined in the NameConstraintsExt module *(Continued)*

Parameter	Description
<code>permittedSubtrees<n>.base.generalNameValue</code>	<p>Specifies the general-name value for the permitted subtree you want to include in the extension.</p> <p>Permissible values: Depends on the general-name type you selected in the <code>permittedSubtrees<n>.base.generalNameChoice</code> field.</p> <ul style="list-style-type: none"> • If you selected <code>rfc822Name</code>, the value must be a valid Internet mail address in the <code>local-part@domain</code> format; see the definition of an <code>rfc822Name</code> as defined in RFC 822 (http://www.ietf.org/rfc/rfc0822.txt). You may use upper and lower case letters in the mail address; no significance is attached to the case. For example, <code>testCA@siroe.com</code>. • If you selected <code>directoryName</code>, the value must be a string form of X.500 name, similar to the subject name in a certificate, in the RFC 2253 syntax (see http://www.ietf.org/rfc/rfc2253.txt). Note that RFC 2253 replaces RFC 1779. For example, <code>CN=SubCA, OU=Research Dept, O=SiroeCorp, C=US</code>. • If you selected <code>dNSName</code>, the value must be a valid domain name in the preferred-name syntax as specified by RFC 1034 (http://www.ietf.org/rfc/rfc1034.txt). You may use upper and lower case letters in the domain name; no significance is attached to the case. Do not use the string “ ” for the DNS name. Also don't use the DNS representation for Internet mail addresses; such identities should be encoded as <code>rfc822Name</code>. For example, <code>testCA.siroe.com</code>. • If you selected <code>ediPartyName</code>, the value must be a <code>IA5String</code>. For example, <code>Siroe Corporation</code>. • If you selected <code>URL</code>, the value must be a non-relative universal resource identifier (URI) following the URL syntax and encoding rules specified in RFC 1738. That is, the name must include both a scheme (for example, <code>http</code>) and a fully qualified domain name or IP address of the host. For example, <code>http://testCA.siroe.com</code>.

Table 4-16 Description of parameters defined in the NameConstraintsExt module *(Continued)*

Parameter	Description
	<ul style="list-style-type: none">If you selected <code>ipAddress</code>, the value must be a valid IP address (IPv4 or IPv6) specified in the dot-separated numeric component notation. The syntax for specifying the IP address is as follows: For IP version 4 (IPv4), the address should be in the form specified in RFC 791 (http://www.ietf.org/rfc/rfc0791.txt). IPv4 address must be in the <code>n.n.n.n</code> format; for example, <code>128.21.39.40</code>. IPv4 address with netmask must be in the <code>n.n.n.n,m.m.m.m</code> format. For example, <code>128.21.39.40,255.255.255.00</code>. For IP version 6 (IPv6), the address should be in the form described in RFC 1884 (http://www.ietf.org/rfc/rfc1884.txt), with netmask separated by a comma. Examples of IPv6 addresses with no netmask are <code>0:0:0:0:0:0:13.1.68.3</code> and <code>FF01::43</code>. Examples of IPv6 addresses with netmask are <code>0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0</code> and <code>FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000</code>.If you selected <code>oid</code>, the value must be a unique, valid OID specified in dot-separated numeric component notation. Although you can invent your own OIDs for the purposes of evaluating and testing this server, in a production environment, you should comply with the ISO rules for defining OIDs and for registering subtrees of IDs. See Appendix B, “Object Identifiers” for information on allocating private OIDs. For example, <code>1.2.3.4.55.6.5.99</code>.If you selected <code>otherName</code>, the value must be the absolute path to the file that contains the base-64 encoded string of the subtree. For example, <code>/opt/SUNWcertsrv/certsrv47/ext/nc/othername.txt</code>.
<code>permittedSubtrees<n>.min</code>	<p>Specifies the minimum number of permitted subtrees.</p> <p>Permissible values: -1, 0, or <code>n</code>.</p> <ul style="list-style-type: none">-1 specifies that the field should not be set in the extension.0 specifies that the minimum number of subtrees is zero (default).<code>n</code> must be an integer that is greater than zero. It specifies at the most <code>n</code> subtrees are allowed. <p>Example: 0</p>

Table 4-16 Description of parameters defined in the NameConstraintsExt module *(Continued)*

Parameter	Description
<code>permittedSubtrees<n>. max</code>	<p>Specifies the maximum number of permitted subtrees.</p> <p>Permissible values: -1, 0, or n.</p> <ul style="list-style-type: none">-1 specifies that the field should not be set in the extension (default).0 specifies that the maximum number of subtrees is zero.n must be an integer that is greater than zero. It specifies at the most n subtrees are allowed. <p>Example: 1</p>
<code>excludedSubtrees<n>. base.generalNameChoice</code>	<p>Specifies the general-name type for the excluded subtree you want to include in the extension.</p> <p>Permissible values: <code>rfc822Name</code>, <code>directoryName</code>, <code>dNSName</code>, <code>ediPartyName</code>, <code>URL</code>, <code>iPAddress</code>, <code>OID</code>, or <code>otherName</code>.</p> <ul style="list-style-type: none">Select <code>rfc822Name</code> if the subtree is an Internet mail address.Select <code>directoryName</code> if the subtree is an X.500 directory name.Select <code>dNSName</code> if the subtree is a DNS name.Select <code>ediPartyName</code> if the subtree is a EDI party name.Select <code>URL</code> if the subtree is a uniform resource locator.Select <code>iPAddress</code> if the subtree is an IP address.Select <code>OID</code> if the subtree is an object identifier.Select <code>otherName</code> if the subtree is in any other name form. <p>Example: <code>OID</code></p>
<code>excludedSubtrees<n>. base.generalNameValue</code>	<p>Specifies the general-name value for the excluded subtree you want to include in the extension.</p> <p>Permissible values: Depends on the general-name type you selected in the <code>excludedSubtrees<n>.base.generalNameChoice</code> field.</p> <ul style="list-style-type: none">If you selected <code>rfc822Name</code>, the value must be a valid Internet mail address in the <code>local-part@domain</code> format; see the definition of an <code>rfc822Name</code> as defined in RFC 822 (http://www.ietf.org/rfc/rfc0822.txt). You may use upper and lower case letters in the mail address; no significance is attached to the case. For example, <code>testCA@siroe.com</code>.

Table 4-16 Description of parameters defined in the NameConstraintsExt module (*Continued*)

Parameter	Description
	<ul style="list-style-type: none"> If you selected <code>directoryName</code>, the value must be a string form of X.500 name, similar to the subject name in a certificate, in the RFC 2253 syntax (see http://www.ietf.org/rfc/rfc2253.txt). Note that RFC 2253 replaces RFC 1779. For example, <code>CN=SubCA,OU=Research Dept,O=Siroe Corp,C=US</code>. If you selected <code>dnsName</code>, the value must be a valid domain name in the preferred-name syntax as specified by RFC 1034 (http://www.ietf.org/rfc/rfc1034.txt). You may use upper and lower case letters in the domain name; no significance is attached to the case. Do not use the string “ ” for the DNS name. Also don’t use the DNS representation for Internet mail addresses; such identities should be encoded as <code>rfc822Name</code>. For example, <code>testCA.siroe.com</code>. If you selected <code>ediPartyName</code>, the value must be an IA5String. For example, <code>Siroe Corporation</code>. If you selected <code>URL</code>, the value must be a non-relative universal resource identifier (URI) following the URL syntax and encoding rules specified in RFC 1738. That is, the name must include both a scheme (for example, <code>http</code>) and a fully qualified domain name or IP address of the host. For example, <code>http://testCA.siroe.com</code>. If you selected <code>ipAddress</code>, the value must be a valid IP address (IPv4 or IPv6) specified in the dot-separated numeric component notation. The syntax for specifying the IP address is as follows: For IP version 4 (IPv4), the address should be in the form specified in RFC 791 (http://www.ietf.org/rfc/rfc0791.txt). IPv4 address must be in the <code>n.n.n.n</code> format; for example, <code>128.21.39.40</code>. IPv4 address with netmask must be in the <code>n.n.n.n,m.m.m.m</code> format. For example, <code>128.21.39.40,255.255.255.00</code>. For IP version 6 (IPv6), the address should be in the form described in RFC 1884 (http://www.ietf.org/rfc/rfc1884.txt), with netmask separated by a comma. Examples of IPv6 addresses with no netmask are <code>0:0:0:0:0:0:13.1.68.3</code> and <code>FF01::43</code>. Examples of IPv6 addresses with netmask are <code>0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0</code> and <code>FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000</code>. If you selected <code>OID</code>, the value must be a unique, valid OID specified in dot-separated numeric component notation. For example, <code>1.2.3.4.55.6.5.99</code>.

Table 4-16 Description of parameters defined in the NameConstraintsExt module *(Continued)*

Parameter	Description
	<ul style="list-style-type: none">If you selected <code>otherName</code>, the value must be the absolute path to the file that contains the base-64 encoded string of the subtree. For example, <code>/opt/SUNWcertsrv/certsrv47/ext/nc/othername.txt</code>.
<code>excludedSubtrees<n>.min</code>	<p>Specifies the minimum number of excluded subtrees.</p> <p>Permissible values: -1, 0, or n.</p> <ul style="list-style-type: none">-1 specifies that the field should not be set in the extension.0 specifies that the minimum number of subtrees is zero (default).n must be an integer that is greater than zero. It specifies at the most n subtrees are allowed. <p>Example: 0</p>
<code>excludedSubtrees<n>.max</code>	<p>Specifies the maximum number of excluded subtrees.</p> <p>Permissible values: -1, 0, or n.</p> <ul style="list-style-type: none">-1 specifies that the field should not be set in the extension (default).0 specifies that the maximum number of subtrees is zero.n must be an integer that is greater than zero. It specifies at the most n subtrees are allowed. <p>Example: 1</p>

NameConstraintsExt Rule

The policy rule named `NameConstraintsExt` is an instance of the `NameConstraintsExt` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is disabled; for the rule to be effective, it must be enabled and configured appropriately.
- The predicate expression is set (`predicate=HTTP_PARAMS.certType==ca`) so that the extension gets added to CA certificates only.
- The extension is marked critical (to comply with the PKIX recommendation).
- The total number of permitted subtrees to be contained in the extension is set to 3 (`numPermittedSubtrees=3`).

- The total number of excluded subtrees to be contained in the extension is set to 3 (`numExcludedSubtrees=3`).
- The maximum number of permitted subtrees is set to -1 (`permittedSubtrees<n>.max=-1`) and the minimum number of permitted subtrees is set to 0 (`permittedSubtrees<n>.min=0`).
- The maximum number of excluded subtrees is set to -1 (`excludedSubtrees<n>.max=-1`) and the minimum number of excluded subtrees is set to 0 (`excludedSubtrees<n>.min=0`).
- The general name type and value fields for each subtree are left blank for you to select or enter the appropriate values.

For details on individual parameters defined in the rule, see Table 4-16 on page 204. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

NSCCommentExt Plug-in Module

The `NSCCommentExt` plug-in module implements the Netscape certificate comment extension policy. This policy enables you to configure Certificate Management System to add the *Netscape Certificate Comment Extension* (see <http://www.netscape.com/eng/security/cert-exts.html>) to certificates. The extension can be used to include textual comments in certificates. Applications that are capable of interpreting the comment may display it to a relying party when the certificate is used or viewed.

For general guidelines on setting the Netscape certificate comment extension, see “netscape-comment” on page 371.

The Netscape certificate comment extension policy in Certificate Management System allows you to specify a textual statement or a comment to be included in certificates. You may choose to directly embed the text in the certificate itself or point to the file that contains the statement.

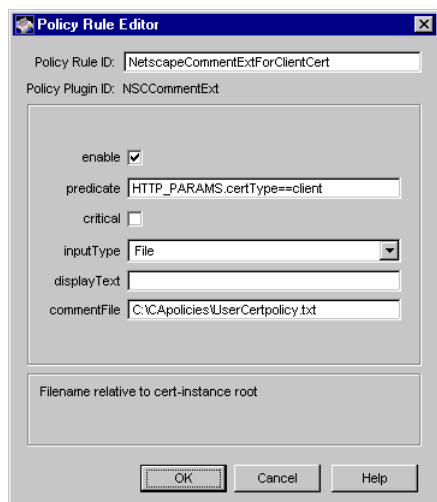
During installation, Certificate Management System automatically creates an instance of the Netscape certificate comment extension policy. See “NSCCommentExt Rule” on page 214.

Configuration Parameters of NSCCommentExt

In the CMS configuration file, the `NSCCommentExt` module is identified as `<subsystem>.Policy.impl.NSCCommentExt.class=com.netscape.certsrv.policy.NSCCommentExt`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `NSCCommentExt`. Figure 4-19 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-19 Parameters defined in the NSCCommentExt module



The configuration shown in Figure 4-19 creates a policy rule named `NetscapeCommentExtForClientCert`, which enforces a rule that the server should set the Netscape certificate comment extension in client certificates.

Table 4-17 provides details for each of these parameters.

Table 4-17 Description of parameters defined in the NSCCommentExt module

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule. Uncheck the box to disable the rule (default).</p> <ul style="list-style-type: none"> If you enable the rule and set the remaining parameters correctly, the server adds the Netscape certificate comment extension to certificates specified by the <code>predicate</code> parameter. If you enable the policy without specifying values in <code>displayText</code> and <code>commentfile</code> fields, the server puts an empty string in the comment extension. If you disable the rule, the server does not add the extension to certificates; it ignores the values in the remaining fields.
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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client</code></p>
critical	<p>Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).</p>
inputType	<p>Specifies whether to embed a textual statement or to include a pointer to file that contains the textual statement in certificates. The extension value is interpreted according to the value specified for this parameter.</p> <p>You should consider putting the textual statement in a file because certain applications may not have the capability to display the text embedded in certificates. Also, embedding a textual statement in a certificate increases its size. If you’re using smart cards for generating and storing certificates, you may not want to embed textual statements in certificates because on a smart card the memory for a certificate may be limited.</p> <p>Permissible values: Text or File.</p> <ul style="list-style-type: none"> <code>Text</code> specifies that the textual statement—the value of the <code>displayText</code> field—should be inserted in the extension (default). <code>File</code> specifies that the path to the file that contains the textual statement—the value of the <code>commentfile</code> field—should be inserted in the extension. <p>Example: <code>File</code></p>

Table 4-17 Description of parameters defined in the NSCCommentExt module *(Continued)*

Parameter	Description
displayText	<p>Specifies the textual statement that should be included in certificates. If you want to embed a textual statement (for example, your company's legal notice) in certificates, then add that statement here. The text you enter here will be displayed to a relying party when the certificate is used or viewed.</p> <p>Permissible values: A string with up to 200 characters.</p> <p>Example: SiroeCorp's CPS incorp. by reference liab. ltd. (c)99 SiroeCorp</p>
commentfile	<p>Specifies the path to the file that contains the textual statement that should be included in certificates; be sure to include the complete path, including the filename. Note that the existence of the file is not checked at the time of policy configuration. The filename will be checked when the policy is applied to a request.</p> <p>Example: C:\CApolicies\UserCertpolicy.txt</p>

NSCCommentExt Rule

The policy rule named `NSCCommentExt` is an instance of the `NSCCommentExt` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is disabled; for the rule to be effective, it must be enabled and configured appropriately.
- The `predicate` field is left blank so that the extension gets added to all certificates.
- The extension is marked noncritical.
- The textual statement is to be embedded in certificates (`inputType=Text`).
- The `displayText` and `commentfile` fields are left blank for you to enter the appropriate information.

For details on individual parameters defined in the rule, see Table 4-17 on page 213. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

NSCertTypeExt Plug-in Module

The `NSCertTypeExt` plug-in module implements the Netscape certificate type extension policy. This policy enables you to configure Certificate Management System to add the *Netscape Certificate Type* extension to certificates. The extension identifies the certificate type—for example, it identifies whether the certificate is a CA certificate, server SSL certificate, client SSL certificate, object signing certificate, or S/MIME certificate—and thus enables you to restrict the usage of a certificate to predetermined purposes.

- If the extension exists in a certificate, it limits the uses of the certificate to those specified (it limits the applications for a certificate).
- If the extension is not present, the certificate can be used for all applications except object signing.

The Netscape certificate type extension is a string of boolean bit-flags, each bit identifying the purpose for which a certificate to be used. Table 4-18 lists the bits and their designated purposes. The extension has no default value.

Table 4-18 Netscape certificate type extension bits and designated purposes

Bit	Purpose	Description
0	SSL Client	Specifies that the certificate can be used by clients for authentication during SSL connections.
1	SSL Server	Specifies that the certificate can be used by servers for authentication during SSL connections.
2	S/MIME	Specifies that the certificate can be used to send secure email messages.
3	Object Signing	Specifies that the certificate can be used for signing objects such as Java applets and plug-ins.
4	Reserved	This bit is reserved for future use.
5	SSL CA	Specifies that the certificate can be used by a CA to issue certificates for SSL connections.
6	S/MIME CA	Specifies that the certificate can be used by a CA to issue certificates for secure email.
7	Object Signing CA	Specifies that the certificate can be used by a CA to issue certificates for object signing.

The Netscape certificate type extension policy has been implemented in such a way that it enables you to set the appropriate certificate-type bits for certificates being issued by Certificate Management System. This way, you can restrict the purposes for which a certificate should be used by adding the extension, with the appropriate bits set, to the certificate at the time of issuance. For example, if you want to restrict a certificate to be used for SSL client authentication only, when issuing the certificate you would add the Netscape certificate type extension to the certificate with `ssl_client` (bit 0) set. For general guidelines on setting the Netscape certificate type extension, see “netscape-cert-type” on page 370.

In the current implementation, you can specify whether to add the extension to certificates on the server side and which bits in the extension are to be set on the client side—you specify whether to add the extension by enabling the Netscape certificate type extension policy and which bits are to be set by adding the appropriate HTTP variables to the enrollment forms.

Bits set in the Netscape certificate type extension are formed from pre-defined input variables that you can embed as hidden values in the default enrollment forms (see “Enrollment Forms” on page 57). Table 4-19 lists the HTTP input variables that correspond to Netscape certificate type extension bits.

Table 4-19 HTTP input variables for Netscape certificate type extension bits

HTTP input variable	Netscape certificate type extension bit
ssl_client	SSL Client (bit 0)
ssl_server	SSL Server (bit 1)
email	S/MIME (bit 2)
object_signing	Object Signing (bit 3)
	Reserved for future use (bit 4)
ssl_ca	SSL CA (bit 5)
email_ca	S/MIME CA (bit 6)
object_signing_ca	Object Signing CA (bit 7)

During installation, Certificate Management System automatically creates an instance of the Netscape certificate type extension policy for the various types of certificates that you may want the server to issue. See “NSCertTypeExt Rule” on page 220.

Additionally, the default enrollment forms—the directory-based, directory- and PIN-based, manual, Kerberos server-based, and NIS server-based enrollment forms—for various types of certificates also include the appropriate HTTP input variables corresponding to Netscape certificate type extension bits. For details about these forms, see “Enrollment Forms” on page 57.

Figure 4-20 shows the default directory-based enrollment form for end users with HTTP input variables specific to Netscape certificate type extension highlighted; it shows two of the total number of variables listed in Table 4-19, `ssl_client` and `email`, indicating that these bits be set in certificates requested using this form.

Figure 4-20 Netscape certificate type extension-specific variables in enrollment forms



```
VIM - /export/cms/bin/cert/web/ee/DirUserEnroll.html
Window Edit Options Help

<font size="-1" face="PrimaSans BT, Verdana, Arial, Helvetica, sans-serif">Pas
sword: </font>
</div>
</td>
<td valign="TOP">
  <input type="PASSWORD" name="pwd" AutoComplete=off size="30">
</td>
</tr>
<tr>
</tr>
</tr>
</table>

<table border="0" width="100%" cellpadding="2" cellspacing="2">
<tr>
<td valign="TOP">
  <!-- for Netscape Certificate Type Extension -->
  <input type="HIDDEN" name="email" value="true">
  <input type="HIDDEN" name="ssl_client" value="true">
  <!-- for Key Usage Extension -->
  <input type="HIDDEN" name="digital_signature" value=true>
  <input type="HIDDEN" name="non_repudiation" value=true>
  <input type="HIDDEN" name="key_encipherment" value=true>
```

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 the bit settings, be sure to add or remove the corresponding variable. Also, when adding a new variable, make sure that the HTML input format is as follows:

```
<input type="HIDDEN" value="true" name="variable_name">
```

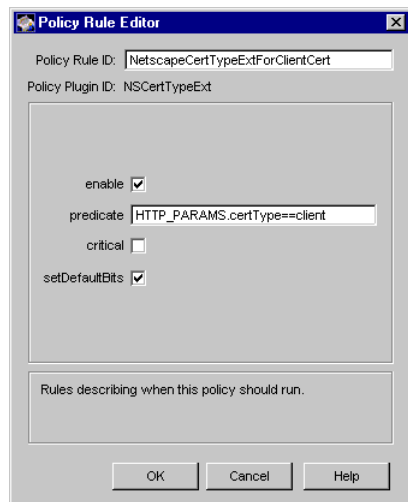
where `variable_name` can be any of the variables listed in Table 4-19.

Configuration Parameters of NSCertTypeExt

In the CMS configuration file, the `NSCertTypeExt` module is identified as `<subsystem>.Policy.impl.NSCertTypeExt.class=com.netscape.certsrv.policy.NSCertTypeExt`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `NSCertTypeExt`. Figure 4-21 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-21 Parameters defined in the NSCertTypeExt module



The configuration shown in Figure 4-21 creates a policy rule named `NetscapeCertTypeExtForClientCert`, which enforces a rule that the server should set the Netscape certificate type extension in client certificates.

Table 4-20 gives details about each of these parameters.

Table 4-20 Description of parameters defined in the NSCertTypeExt module

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule. Uncheck the box to disable the rule (default).</p> <ul style="list-style-type: none"> If you enable the rule, be sure to review the enrollment forms for Netscape certificate type extension-specific variables and to set the remaining parameters of this policy correctly. If the bits are unspecified in the enrollment form, the server checks the value assigned to the <code>setDefaultBits</code> parameter. If it is unchecked (false), the server does not set the extension. If you disable the rule, the server does not add the extension to certificates; it ignores the Netscape certificate type extension-specific HTTP input values in the certificate request and the status of the <code>setDefaultBits</code> parameter.
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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client</code></p>
setDefaultBits	<p>Specifies whether to set the Netscape certificate type extension with default bits in certificates specified by the predicate expression.</p> <ul style="list-style-type: none"> Check the box the if you want the server to add the extension, with default bits, to certificates. If you check the box and if no bits are requested from the HTTP input, the server adds the Netscape certificate type extension to certificates with the following bits set: <ul style="list-style-type: none"> - <code>ssl client</code> (bit 0) - <code>email</code> (bit 2) Uncheck the box if you don’t want the server to add the extension with default bits. If you uncheck the box and if no bits are requested from the HTTP input, the server does not add the extension to certificates.

NSCertTypeExt Rule

The policy rule named `NSCertTypeExt` is an instance of the `NSCertTypeExt` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression is set so that the extension gets added to all certificates except the ones issued to routers
(predicate=`HTTP_PARAMS.certType!=CEP-Request`).
- The server sets the default bits if the bits are unspecified in the enrollment form.

For details on individual parameters defined in the rule, see Table 4-20 on page 219. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

OCSPNoCheckExt Plug-in Module

The `OCSPNoCheckExt` plug-in module implements the OCSF no check extension policy. This policy enables you to configure Certificate Management System to add the *OCSP No Check Extension* defined in X.509 and PKIX standard RFC 2560 (see <http://www.ietf.org/rfc/rfc2560.txt>) to certificates. The extension, which should be used in OCSF responder certificates only, indicates how OCSF-compliant applications can verify the revocation status of the certificate an authorized OCSF responder uses to sign OCSF responses.

The online certificate status protocol (OCSF) enables OCSF-compliant applications to determine the revocation status of a certificate being validated. Certificate Management System supports the OCSF service—you can configure a Certificate Manager to publish CRLs to an online validation authority, also called OCSF responder (see Chapter 21, “Setting Up an OCSF Responder” of *CMS Installation and Setup Guide*). If you configure Certificate Management System to work with an OCSF responder, OCSF-compliant applications in your PKI setup will be able to do real-time verification of certificates by querying the OCSF responder for their revocation status. Note that these applications will be able to query the OCSF

responder only if the certificate being validated includes the authority information access extension indicating the location of the OCSP responder; for information on adding this extension to certificates, see “AuthInfoAccessExt Plug-in Module” on page 136.

When queried by an application on the status of a certificate, the OCSP responder sends a digitally signed response. For the signature, the responder uses the key pair designated for signing OCSP responses. Usually, the CA issues an *OCSP responder certificate* to the responder, which enables applications to identify it as a CA-designated responder. The CA issues this certificate with an extended key usage extension with a unique value, which indicates that the key associated with the certificate can be used for signing OCSP responses. For details on this extension, see “OCSPSigningExt Rule” on page 176.

When an OCSP-compliant application receives a signed response, as a part of validating the signature, the application needs to verify that the responder’s certificate has not been revoked. RFC 2560 recommends three ways in which a CA may indicate the revocation status of an OCSP responder certificate. One of them is that the CA issue the OCSP responder a certificate with the OCSP no check extension, which indicates that the certificate can be trusted by the clients for its lifetime. The OCSP no check policy of Certificate Management System implements this method and enables you to set the OCSP no check extension in OCSP responder certificates.

Because OCSP-compliant applications don’t check for the revocation status of the OCSP responder certificate (containing the OCSP no check extension), when issuing these types of certificates, you should consider issuing them with a short validity period (and renew them frequently). Note that the OCSP no check extension policy only adds the extension to a certificate; it doesn’t control the validity period of the certificate. If you want to limit the validity period of these certificates to a short period, you should consider creating an instance of the `ValidityConstraints` module with the appropriate configuration, for example, set the predicate parameter to `HTTP_PARAMS.certType=ocspResponder`. For details, see “ValidityConstraints Plug-in Module” on page 124. If you have agent privileges, you can also specify the required validity period when approving the OCSP responder certificate request in the request queue; the enrollment process for an OCSP responder certificate is manual, and the request gets queued for agent approval.

Before configuring the server to add the OCSP no check extension to OCSP responder certificates, read the general guidelines provided in “OCSPNocheck” on page 354.

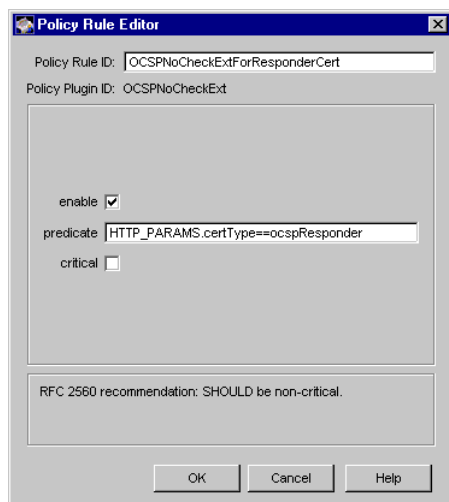
During installation, Certificate Management System automatically creates an instance of the OCSP no check extension policy. See “OCSPNoCheckExt Rule” on page 223.

Configuration Parameters of OCSPNoCheckExt

In the CMS configuration file, the OCSPNoCheckExt module is identified as `<subsystem>.Policy.impl.OCSPNoCheckExt.class=com.netscape.certsrv.policy.OCSPNoCheckExt`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as OCSPNoCheckExt. Figure 4-22 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-22 Parameters defined in the OCSPNoCheckExt module



The configuration shown in Figure 4-22 creates a policy rule named OCSPNoCheckExtForResponderCert, which enforces a rule that the server should set the OCSP no check extension in OCSP responder certificates only.

Table 4-21 provides details for each of these parameters.

Table 4-21 Description of parameters defined in the OCSPNoCheckExt module

Parameter	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none"> If you enable the rule and set the remaining parameters correctly, the server adds the OCSP no check extension to certificates specified by the <code>predicate</code> parameter. If you disable the rule, the server does not add the extension to certificates; it ignores the values in the remaining fields.
<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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==ocspResponder</code></p>
<code>critical</code>	<p>Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).</p>

OCSPNoCheckExt Rule

The policy rule named `OCSPNoCheckExt` is an instance of the `OCSPNoCheckExt` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression is set (`predicate=HTTP_PARAMS.certType==ocspResponder`) so that the extension gets added to OCSP responder certificates only.
- The extension is marked noncritical (to comply with the PKIX recommendation).

For details on individual parameters defined in the rule, see Table 4-21 on page 223. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

PolicyConstraintsExt Plug-in Module

The `PolicyConstraintsExt` plug-in module implements the policy constraints extension policy. This policy enables you to configure Certificate Management System to add the *Policy Constraints Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) to certificates. The extension, which can be used in CA certificates only, constrains path validation in two ways—either to prohibit policy mapping or to require that each certificate in a path contain an acceptable policy identifier.

The policy constraints extension policy in Certificate Management System allows setting of the policy constraints extension as defined in its X.509 definition. The policy allows you to specify both, `requireExplicitPolicy` and `inhibitPolicyMapping` fields. PKIX standard requires that, if present in a CA certificate, the extension must never consist of a null sequence. At least one of the two specified fields must be present. Before configuring the server to add the policy constraints extension to certificates, read the general guidelines provided in “policyConstraints” on page 355.

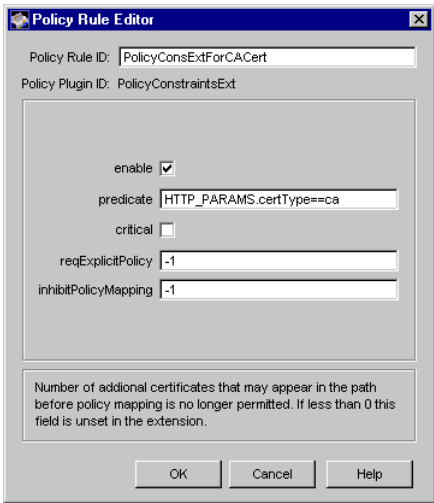
During installation, Certificate Management System automatically creates an instance of the policy constraints extension policy. See “PolicyConstraintsExt Rule” on page 227.

Configuration Parameters of PolicyConstraintsExt

In the CMS configuration file, the `PolicyConstraintsExt` module is identified as `ca.Policy.impl.PolicyConstraintsExt.class=com.netscape.certsrv.policy.PolicyConstraintsExt`.

In the CMS window, the module is identified as `PolicyConstraintsExt`. Figure 4-23 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-23 Parameters defined in the PolicyConstraintsExt module



The configuration shown in Figure 4-23 creates a policy rule named `PolicyConsExtForCACert`, which enforces a rule that the server should set the policy constraints extension in CA certificates only.

Table 4-22 gives details about each of these parameters.

Table 4-22 Description of parameters defined in the PolicyConstraintsExt module

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server adds the policy constraints extension to certificates specified by the <code>predicate</code> parameter.• If you disable the rule, the server does not add the extension to certificates; it ignores the values in the remaining fields.
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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==ca</code></p>

Table 4-22 Description of parameters defined in the PolicyConstraintsExt module *(Continued)*

Parameter	Description
critical	<p>Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).</p>
reqExplicit Policy	<p>Specifies the total number of certificates permitted in the path before an explicit policy is required—that is, the number of CA certificates that can be chained below (subordinate to) the subordinate CA certificate being issued before an acceptable policy is required.</p> <p>Note that the number you specify affects 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>Permissible values: -1, 0, or <i>n</i>.</p> <ul style="list-style-type: none">-1 specifies that the field should not be set in the extension (default).0 specifies that no subordinate CA certificates are permitted in the path before an explicit policy is required.<i>n</i> must be an integer that is greater than zero. It specifies at the most <i>n</i> subordinate CA certificates are allowed in the path before an explicit policy is required. <p>Example: 1</p>
inhibitPolicy Mapping	<p>Specifies the total number of certificates permitted in the path before policy mapping is no longer permitted.</p> <p>Permissible values: -1, 0, or <i>n</i>.</p> <ul style="list-style-type: none">-1 specifies that the field should not be set in the extension (default).0 specifies that no subordinate CA certificates are permitted in the path before policy mapping is no longer permitted.<i>n</i> must be an integer that is greater than zero. It specifies at the most <i>n</i> subordinate CA certificates are allowed in the path before policy mapping is no longer permitted. For example, a value of one indicates that policy mapping may be processed in certificates issued by the subject of this certificate, but not in additional certificates in the path. <p>Example: -1</p>

PolicyConstraintsExt Rule

The policy rule named `PolicyConstraintsExt` is an instance of the `PolicyConstraintsExt` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is disabled; for the rule to be effective, it must be enabled and configured appropriately.
- The predicate expression is set (`predicate=HTTP_PARAMS.certType==ca`) so that the extension gets added to CA certificates only. PKIX and Federal PKI standards recommend that CA certificates must have this extension and end-entity certificates should have this extension.
- The extension is marked noncritical.
- No subordinate CA certificates are permitted in the path before an explicit policy is required (`reqExplicitPolicy=0`).
- The `inhibitPolicyMapping` field is not set in the extension.

For details on individual parameters defined in the rule, see Table 4-22 on page 225. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

PolicyMappingsExt Plug-in Module

The `PolicyMappingsExt` plug-in module implements the policy mappings extension policy. This policy enables you to configure Certificate Management System to add the *Policy Mappings Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) to certificates. The extension lists one or more pairs of OIDs, each pair identifying two policy statements of two CAs. The pairing indicates that the corresponding policies of one CA are equivalent to policies of another CA. The extension may be useful in the context of cross-certification.

The PKIX standard suggests that the extension must be marked noncritical and may be supported by CAs and/or applications. If supported, the extension is to be included in CA certificates only. Before configuring the server to add the policy mappings extension to certificates, read the general guidelines provided in “policyMappings” on page 356.

The policy mappings extension policy in Certificate Management System allows setting of the policy mappings extension as defined in its X.509 definition. The policy allows you to map policy statements of one CA to that of another by pairing the OIDs assigned to their policy statements. (For information on OIDs, see Appendix B, “Object Identifiers.”) For information on certificate policies, see “CertificatePoliciesExt Plug-in Module” on page 151.)

Each pair is defined by two parameters, `issuerDomainPolicy` and `subjectDomainPolicy`. The pairing indicates that the issuing CA considers the `issuerDomainPolicy` equivalent to the `subjectDomainPolicy` of the subject CA. The issuing CA’s users may accept an `issuerDomainPolicy` for certain applications. The policy mapping tells these users which policies associated with the subject CA are equivalent to the policy they accept.

During installation, Certificate Management System automatically creates an instance of the policy mappings extension policy. See “PolicyMappingsExt Rule” on page 231.

Configuration Parameters of PolicyMappingsExt

In the CMS configuration file, the `PolicyMappingsExt` module is identified as `ca.Policy.impl.PolicyMappingsExt.class=com.netscape.certsrv.policy.PolicyMappingsExt`.

In the CMS window, the module is identified as `PolicyMappingsExt`. Figure 4-24 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-24 Parameters defined in the PolicyMappingsExt module



The configuration shown in Figure 4-24 creates a policy rule named `PolicyMapExtForCACert`, which enforces a rule that the server should set the policy mappings extension in CA certificates only.

Table 4-23 provides details for each of these parameters.

Table 4-23 Description of parameters defined in the PolicyMappingsExt module

Parameter	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule. Uncheck the box to disable the rule (default).</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server adds the policy mappings extension to certificates specified by the <code>predicate</code> parameter.• If you disable the rule, the server does not add the extension to certificates; it ignores the values in the remaining fields.
<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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==ca</code></p>
<code>critical</code>	<p>Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).</p>

Table 4-23 Description of parameters defined in the PolicyMappingsExt module *(Continued)*

Parameter	Description
numPolicyMappings	<p>Specifies the total number of policy mapping (pairs) to be contained or allowed in the extension. Note that each policy mapping represents a pair of policies—specified by <code>policyMap<n>.issuerDomainPolicy</code> and <code>policyMap<n>.subjectDomainPolicy</code>—and each policy in the pair belongs to a specific CA.</p> <p>You can change the total number of policy pairs by changing the value assigned to this parameter; there's no restriction on the total number of policy pairs you can include in the extension. Each pair is distinguished by <code><n></code>, which is an integer derived from the value you assign in this field. For example, if you set the <code>numPolicyMappings</code> parameter to 2, <code><n></code> would be 0 and 1.</p> <p>Permissible values: 0 or <code>n</code>.</p> <ul style="list-style-type: none">• 0 specifies that no policy pairs can be contained in the extension.• <code>n</code> specifies the total number of policy pairs to be included in the extension; it must be a integer greater than zero. The default value is 1. <p>Example: 2</p>
policyMap<n>. issuerDomainPolicy	<p>Specifies the OID assigned to the policy statement<<code>n</code>> of the issuing CA that you want to map with the policy statement of another CA.</p> <p>Permissible values: Any valid OID specified in dot-separated numeric component notation (see the example). The OID that you specify should be in the registered subtree of IDs reserved for your company's use. Although you can invent your own OIDs for the purposes of evaluating and testing this server, in a production environment, you should comply with the ISO rules for defining OIDs and for registering subtrees of IDs. See Appendix B, "Object Identifiers" for information on allocating private OIDs.</p> <p>Example: 1.2.3.4.5</p>
policyMap<n>. subjectDomainPolicy	<p>Specifies the OID assigned to the policy statement<<code>n</code>> of the subject CA that corresponds to the policy statement of the issuing CA.</p> <p>Permissible values: Any valid OID specified in dot-separated numeric component notation (see the example).</p> <p>Example: 6.7.8.9.10</p>

PolicyMappingsExt Rule

The rule named `PolicyMappingsExt` is an instance of the `PolicyMappingsExt` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression is set (`predicate=HTTP_PARAMS.certType==ca`) so that the extension gets added to CA certificates only.
- The extension is marked noncritical (to comply with the PKIX recommendation).
- The number of policy mappings is set to 1 (`numPolicyMappings=1`) indicating that a pair of policies are to be mapped.
- The fields for entering the OIDs for policies that are to be mapped are left blank for you to enter the appropriate values.

For details on individual parameters defined in the rule, see Table 4-23 on page 229. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

PrivateKeyUsagePeriodExt Plug-in Module

The `PrivateKeyUsagePeriodExt` plug-in module implements the private key usage period extension policy. This policy enables you to configure Certificate Management System to add the *Private Key Usage Period Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) to certificates. The extension allows the certificate issuer to specify a different validity period for the private key than the one specified for the corresponding certificate. The extension is intended for use with digital signature keys.

The PKIX standard recommends against the use of this extension. The standard also recommends that CAs conforming to the standard must not generate certificates with private key usage period extensions that are marked critical. For general guidelines on setting this extension in certificates, see “privateKeyUsagePeriod” on page 357.

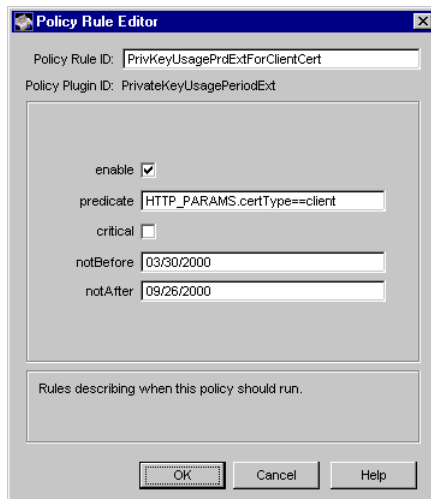
The private key usage period extension policy in Certificate Management System allows setting of the private key usage period extension as defined in its X.509 definition. The policy enables you to specify values for the `notBefore` and `notAfter` components. When included in a certificate, the `notBefore` and `notAfter` components define the time before and after which the private key associated with the certificate should not be used to sign objects.

Configuration Parameters of PrivateKeyUsagePeriodExt

In the CMS configuration file, the `PrivateKeyUsagePeriodExt` module is identified as `<subsystem>.Policy.impl.PrivateKeyUsagePeriodExt.class=com.netscape.certsrv.policy.PrivateKeyUsagePeriodExt`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `PrivateKeyUsagePeriodExt`. Figure 4-25 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-25 Parameters defined in the PrivateKeyUsagePeriodExt module



The configuration shown in Figure 4-25 creates a policy rule named `PrivKeyUsagePrdExtForClientCert`, which enforces a rule that the server should set the private key usage period extension in client certificates.

Table 4-24 provides details for each of these parameters.

Table 4-24 Description of parameters defined in the PrivateKeyUsagePeriodExt module

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule. Uncheck the box to disable the rule (default).</p> <ul style="list-style-type: none"> If you enable the rule and set the remaining parameters correctly, the server adds the private key usage period extension to certificates specified by the <code>predicate</code> parameter. If you disable the rule, the server does not add the extension to certificates; it ignores the values in the remaining fields.
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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client</code></p>
critical	<p>Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).</p>
notBefore	<p>Specifies the date on which the validity period for the private key associated with the certificate begins.</p> <p>Permissible values: A valid date specified in the MM/DD/YYYY format.</p> <p>Example: 12/25/2000</p>
notAfter	<p>Specifies the date on which the validity period for the private key associated with the certificate ends.</p> <p>Permissible values: A valid date specified in the MM/DD/YYYY format.</p> <p>Example: 12/25/2001</p>

RemoveBasicConstraintsExt Plug-in Module

The `RemoveBasicConstraintsExt` plug-in module implements the remove basic constraints extension policy. This policy, if enabled, can detect the presence of *Basic Constraints* extension in a certificate request and remove it. For details about the Basic Constraints extension, see “BasicConstraintsExt Plug-in Module” on page 147.

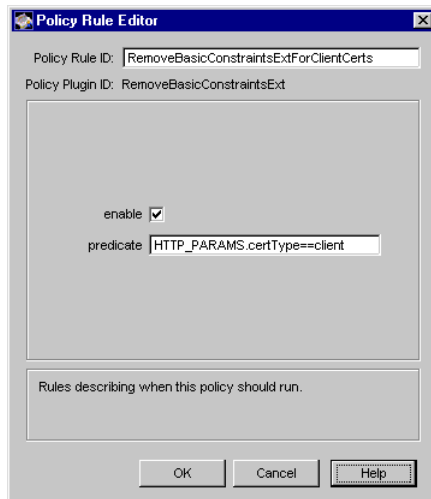
The policy can be useful in certain enrollment scenarios. For example, enrollment requests from customized clients that can generate CRMF requests can include extensions, including the Basic Constraints extension, and the policy can detect the presence of the Basic Constraints extension and remove it.

Configuration Parameters of RemoveBasicConstraintsExt

In the CMS configuration file, the RemoveBasicConstraintsExt module is identified as `ca.Policy.impl.RemoveBasicConstraintsExt.class=com.netscape.certsrv.policy.RemoveBasicConstraintsExt`.

In the CMS window, the module is identified as `RemoveBasicConstraintsExt`. Figure 4-26 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-26 Parameters defined in the RemoveBasicConstraintsExt module



The configuration shown in Figure 4-26 creates a policy rule named `RemoveBasicConstraintsExtForClientCerts`, which enforces a rule that the Basic Constraints extension be removed from all client certificate requests.

Table 4-25 provides details for each of these parameters.

Table 4-25 Description of parameters defined in the RemoveBasicConstraintsExt module

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server checks certificate requests for Basic Constraints extension and removes it.• If you disable the rule, the server does not check the requests for Basic Constraints extension; it ignores the values in the remaining fields.
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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client</code></p>

SubjectAltNameExt Plug-in Module

The SubjectAltNameExt plug-in module implements the subject alternative name policy. This policy enables you to configure Certificate Management System to add the *Subject Alternative Name Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) to certificates. The extension enables you to bind additional identities—such as Internet electronic mail address, a DNS name, an IP address, and a uniform resource indicator (URI)—to the subject of the certificate.

The standard suggests that if the certificate subject field contains an empty sequence, then the subject alternative name extension must contain the subject’s alternative name and that the extension be marked critical. For general guidelines on setting the subject alternate name extension in certificates, see “subjectAltName” on page 358.

The subject alternative name extension policy in Certificate Management System enables you to include values of certificate-request attributes in the extension. You can include any number of attributes as long as the attribute values conform to any of the supported general-name forms: rfc822Name, X.500 directory name, DNS name, EDI party name, URL, IP address, object identifier, and Other name.

Attributes in a certificate request 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, such as the user ID, are set on the request after successful authentication. The servlets can also set additional attributes related to the certificate content on the request; for example, in automated-enrollment methods, some attributes may be read from the authentication directory and set in the request as authenticated attributes.

If you're using any of the directory-based authentication methods, you can configure Certificate Management System to retrieve values for any string and byte attributes from the directory and set them in the certificate request during authentication—you specify these attributes by entering them in the `ldapStringAttributes` and `ldapByteAttributes` fields defined in the automated enrollment modules. For more information, see Table 1-2 on page 28, Table 1-3 on page 32, and Table 1-4 on page 40.

Note that 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.

In general, you can configure which attributes should or shouldn't be stored in the request; for example, you can exclude sensitive attributes such as passwords from getting stored in the request with the help of the parameter named `dontSaveHttpParams` defined in the CMS configuration file. For details on using this parameter, see the description for `HTTP_PARAMS` in section “JavaScript Used By All Interfaces” of *CMS Customization Guide*. You can also distinguish the attributes based on their origin—that is, whether they originated from the enrollment form or where added to the request during the authentication process. Authenticated attributes have `AUTH_TOKEN` as prefix (for example, `AUTH_TOKEN.mail`) and non-authenticated attributes such as the ones that come from the HTTP input have `HTTP_PARAMS` as prefix (for example, `HTTP_PARAMS.csrRequestorEmail`).

If enabled, the subject alternative extension policy checks the certificate request for configured attributes. If the request contains an attribute, the policy reads its value and sets it in the extension. This way, the extension that gets added to certificates contains all the configured attributes.

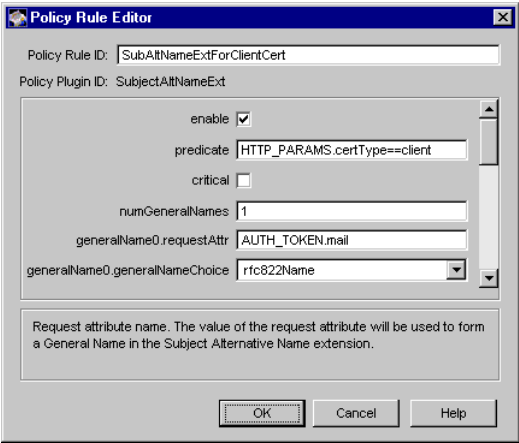
During installation, Certificate Management System automatically creates an instance of the subject alternative name extension policy. See “SubjectAltNameExt Rule” on page 240.

Configuration Parameters of SubjectAltNameExt

In the CMS configuration file, the SubjectAltNameExt module is identified as `<subsystem>.Policy.impl.SubjectAltNameExt.class=com.netscape.certsrv.policy.SubjectAltNameExt`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `SubjectAltNameExt`. Figure 4-27 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-27 Parameters defined in the SubjectAltNameExt module



The configuration shown in Figure 4-27 creates a policy rule named `SubAltNameExtForClientCert`, which enforces a rule that the alternative name of the certificate’s subject must be derived from the `mail` attribute of subject’s entry in the authentication directory and that the server should set the subject alternative name extension only in client certificates.

Table 4-26 provides details for each of these parameters.

Table 4-26 Description of parameters defined in the SubjectAltNameExt module

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server adds the subject alternative name extension to certificates specified by the <code>predicate</code> parameter.• If you disable the rule, the server does not add the extension to certificates; it ignores the values in the remaining fields.
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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client</code></p>
critical	<p>Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).</p>
numGeneralNames	<p>Specifies the total number of alternative names or identities permitted in the extension. Note that each name has a set of configuration parameters—<code>generalName<n>.requestAttr</code> and <code>generalName<n>.generalNameChoice</code>—and you must specify appropriate values for each of those parameters; otherwise the policy rule will return an error.</p> <p>You can change the total number of identities by changing the value of this parameter; there’s no restriction on the total number of identities you can include in the extension. Each set of configuration parameters is distinguished by <code><n></code>, which is an integer derived from the value you assign in this field. For example, if you set the <code>numGeneralNames</code> parameter to 2, <code><n></code> would be 0 and 1.</p> <p>Permissible values: 0 or <code>n</code>.</p> <ul style="list-style-type: none">• 0 specifies that no identities can be contained in the extension.• <code>n</code> specifies the total number of identities to be included in the extension; it must be an integer greater than zero. The default value is 8. <p>Example: 2</p>

Table 4-26 Description of parameters defined in the SubjectAltNameExt module *(Continued)*

Parameter	Description
<code>generalName<n>. requestAttr</code>	<p>Specifies the request attribute whose value is to be included in the extension. The attribute value must conform to any of the supported general-name types (specified by the <code>generalName<n>.generalNameChoice</code> parameter). If the server finds the attribute in the request, it sets the attribute value in the extension and then adds the extension to certificates specified by the <code>predicate</code> parameter. If you specify multiple attributes and if none of the attributes are present in the request, the server does not add the subject alternative name extension to certificates.</p> <p>Permissible values: A request attribute included in the certificate request.</p> <p>Example: <code>AUTH_TOKEN.mail</code></p>
<code>generalName<n>. generalNameChoice</code>	<p>Specifies the general-name type for the request attribute.</p> <p>Permissible values: <code>rfc822Name</code>, <code>directoryName</code>, <code>dNSName</code>, <code>ediPartyName</code>, <code>URL</code>, <code>iPAddress</code>, <code>OID</code>, or <code>otherName</code>.</p> <ul style="list-style-type: none"> • Select <code>rfc822Name</code> if the request-attribute value is an Internet mail address in the <code>local-part@domain</code> format (default). For example, <code>jdoe@siroe.com</code>. • Select <code>directoryName</code> if the request-attribute value is an X.500 directory name, similar to the subject name in a certificate. For example, <code>CN=Jane Doe, OU=Sales Dept, O=Siroe Corp, C=US</code>. • Select <code>dNSName</code> if the request-attribute value is a DNS name. For example, <code>corpDirectory.siroe.com</code>. • Select <code>ediPartyName</code> if the request-attribute value is a EDI party name. For example, <code>Siroe Corporation</code>. • Select <code>URL</code> if the request-attribute value is a non-relative URI that includes both a scheme (for example, <code>http</code>) and a fully qualified domain name or IP address of the host. For example, <code>http://hr.siroe.com</code>. • Select <code>iPAddress</code> if the request-attribute value is a valid IP address specified in dot-separated numeric component notation. For example, <code>128.21.39.40</code>. • Select <code>OID</code> if the request-attribute value is a unique, valid OID specified in the dot-separated numeric component notation. For example, <code>1.2.3.4.55.6.5.99</code>. • Select <code>otherName</code> if the request-attribute value is the absolute path to the file that contains the base-64 encoded string of the subject alternative name. For example, <code>/opt/SUNWcertsrv/certsrv47/ext/san/othername.txt</code>. <p>Example: <code>rfc822Name</code></p>

SubjectAltNameExt Rule

The policy rule named `SubjectAltNameExt` is an instance of the `SubjectAltNameExt` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression is left blank so that the extension gets added to all certificates the server issues. (PKIX and Federal PKI standards recommend that CA certificates must have this extension and end-entity certificates should have this extension.)
- The extension is marked noncritical (to comply with the PKIX recommendation).
- The rule is configured to include at the most three alternative names in the extension (`numGeneralNames=3`).
- The first alternative name is the value of the `mail` attribute in the certificate subject's directory entry (`generalName0.requestAttr=AUTH_TOKEN.mail`) and the name is in the `rfc822Name` format (`generalName0.generalNameChoice=rfc822Name`).
- The second alternative name is the value of the `mailalternateaddress` attribute in the certificate subject's directory entry (`generalName1.requestAttr=AUTH_TOKEN.mailalternateaddress`) and the name is in the `rfc822Name` format (`generalName1.generalNameChoice=rfc822Name`).
- The third alternative name is the value of an HTTP input parameter `csrRequestorEmail` included in the certificate request (`generalName2.requestAttr=HTTP_PARAMS.csrRequestorEmail`) and the name is in `rfc822Name` format (`generalName2.generalNameChoice=rfc822Name`).

For details on individual parameters defined in the rule, see Table 4-26 on page 238. You need to review this rule and make the changes appropriate for your PKI setup. For instructions, see section “Step 2. Modify Existing Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section “Step 4. Add New Policy Rules” in the same chapter.

Before you edit the default rule, you should read the additional details about the attributes that are set in the default policy rule.

The first two attributes, `AUTH_TOKEN.mail` and `AUTH_TOKEN.mailalternateaddress`, are standard LDAP attributes typically used for storing end users' email addresses in an LDAP directory. These attributes enable you to include a user's email address as an alternative name in the certificate. Remember that you need to specify the LDAP attribute for users' email addresses as a part of configuring the server to use a specific directory for authentication—which means for the default rule to set end users' email addresses in the subject alternative name extension, you must ensure the following:

- The server is configured for directory-based, directory- and PIN-based, or NIS server based (using directory attributes for forming subject names) enrollment; that is, you have created and configured an authentication instance.
- The `ldapStringAttributes` parameter in the authentication instance is set to `mail` or `mailalternateaddress`, or to both.

The third attribute, `HTTP_PARAMS.csrRequestorEmail`, is the email component of the subject name in an enrollment request—it is an HTTP input value that gets added to the request when a user uses the *manual* enrollment form; for details, see “Enrollment Forms” on page 57.

If you enable the default policy rule, the server automatically checks the certificate request for attributes `AUTH_TOKEN.mail`, `AUTH_TOKEN.mailalternateaddress`, and `HTTP_PARAMS.csrRequestorEmail`. If the server finds any of the attributes, it sets the attribute value in the extension and then adds the extension to certificates specified by the `predicate` parameter. If none of the attributes are in a request, the server does not add the subject alternative name extension to the certificate.

SubjectDirectoryAttributesExt Plug-in Module

The `SubjectDirectoryAttributesExt` plug-in module implements the subject directory attributes extension policy. This policy enables you to configure Certificate Management System to add the *Subject Directory Attributes Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) to certificates. The extension is used to specify any desired directory attribute values for the subject of the certificate.

As per the PKIX standard, inclusion of this extension in certificates is not essential; the standard suggests that the extension may be used in local environments. For general guidelines on setting the subject directory attributes extension, see “subjectDirectoryAttributes” on page 359.

The subject directory attributes extension policy in Certificate Management System allows you to include up to three directory attributes in the extension. For each attribute that you want to include in the extension, you need to specify the attribute name and its value—the name must be the X.500 directory attribute name itself and the attribute value can be derived from the request or directly entered in the policy configuration as a string value.

The list of directory attributes supported by default are shown as permissible values for the `attribute<n>.attributeName` parameter explained in Table 4-27 on page 243. You can extend the list of attributes supported by the policy by defining new X.500 directory attributes. For details on defining new attributes, see “Extending Attribute Support” on page 318.

Note that, during installation, Certificate Management System does not create an instance of the subject directory attributes extension policy. If you want the server to add this extension to certificates, you must create an instance of the `SubjectDirectoryAttributesExt` module and configure it. For instructions, see section “Step 4. Add New Policy Rules” in Chapter 18, “Setting Up Policies” of *CMS Installation and Setup Guide*.

Configuration Parameters of SubjectDirectoryAttributesExt

In the CMS configuration file, the `SubjectDirectoryAttributesExt` module is identified as `<subsystem>.Policy.impl.SubjectDirectoryAttributesExt.class=com.netscape.certsrv.policy.SubjectDirectoryAttributesExt`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `SubjectDirectoryAttributesExt`. Figure 4-28 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-28 Parameters defined in the SubjectDirectoryAttributesExt module

The screenshot shows the 'Policy Rule Editor' dialog box. At the top, 'Policy Rule ID' is 'SubDirAttrForClientCert' and 'Policy Plugin ID' is 'SubjectDirectoryAttributesExt'. The 'enable' checkbox is checked. The 'predicate' field is empty. The 'critical' checkbox is unchecked. The 'numAttributes' field is '2'. There are three attribute configurations: 1. 'attribute0.attributeName' is 'CN', 'attribute0.whereToGetValue' is 'Request Attribute', and 'attribute0.value' is 'HTTP_PARAMS.CN'. 2. 'attribute1.attributeName' is 'STREET', 'attribute1.whereToGetValue' is 'Fixed Value', and 'attribute1.value' is '501 E. Middlefield Rd'. 3. 'attribute2.attributeName' is 'TITLE', 'attribute2.whereToGetValue' is 'Request Attribute', and 'attribute2.value' is empty. A note at the bottom states: 'Adds Subject Directory Attributes extension. See RFC 2459 (4.2.1.9). It's not recommended as an essential part of the profile, but may be used in local environments.' Buttons for 'OK', 'Cancel', and 'Help' are at the bottom.

The configuration shown in Figure 4-28 creates a policy rule named `SubDirAttrForClientCert`, which enforces a rule that the server should set the subject directory attributes extension in client certificates.

Table 4-27 provides details for each of these parameters.

Table 4-27 Description of parameters defined in the SubjectDirectoryAttributesExt module

Parameter	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule. Uncheck the box to disable the rule (default).</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server adds the subject directory attributes extension to certificates specified by the <code>predicate</code> parameter.• If you disable the rule, the server does not add the extension to certificates; it ignores the values in the remaining fields.

Table 4-27 Description of parameters defined in the SubjectDirectoryAttributesExt module *(Continued)*

Parameter	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 (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i>.</p> <p>Example: <code>HTTP_PARAMS.certType==client AND HTTP_PARAMS.OU==Engineering</code></p>
<code>critical</code>	<p>Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).</p>
<code>numAttributes</code>	<p>Specifies the total number of directory attributes to be contained or allowed in the extension. Note that each attribute has a name (or OID) and value and you must specify appropriate values for both; otherwise the policy rule will return an error.</p> <p>You can configure the server to include up to three attributes in the extension. By default, this field is set to its maximum value, 3, and the UI shows fields for configuring three attributes. You can change the total number of attributes by changing the value of this parameter. Each set of configuration parameters is distinguished by <code><n></code>, which is an integer derived from the value you assign in this field. For example, if you set the <code>numAttributes</code> parameter to 2, <code><n></code> would be 0 and 1.</p> <p>Permissible values: 1, 2, or 3. The default value is 3.</p> <p>Example: 1</p>
<code>attribute<n>.attributeName</code>	<p>Specifies the name of the directory attribute whose value is to be included in the extension.</p> <p>Permissible values: TITLE, O, OU, L, E, C, GIVENNAME, DC, UID, CN, UNSTRUCTUREDNAME, GENERATIONQUALIFIER, ST, DNQUALIFIER, SN, MAIL, UNSTRUCTUREDADDRESS, STREET, SERIALNUMBER, and INITIALS. The list may show any additional attributes that you may have added.</p> <p>Example: TITLE</p>

Table 4-27 Description of parameters defined in the SubjectDirectoryAttributesExt module *(Continued)*

Parameter	Description
<code>attribute<n>.whereToGetValue</code>	<p>Specifies from where to get the value for the selected directory attribute.</p> <p>Permissible values: <code>Request Attribute</code> or <code>Fixed Value</code>.</p> <ul style="list-style-type: none"> • Select <code>Request Attribute</code> if you want the server to read the value from the request attribute. • Select <code>Fixed Value</code> if you want to specify a fixed value for the attribute. • Note that both the options require you to enter the value for the attribute in the <code>attribute<n>.value</code> field. The server will set the extension with this value in all certificates specified by the <code>predicate</code> parameter. <p>Example: <code>Fixed Value</code></p>
<code>attribute<n>.value</code>	<p>Specifies the value for the directory attribute to be included in the extension.</p> <p>Permissible value: A string value for the attribute selected.</p> <p>Example: <code>Member of Technical Staff</code></p>

SubjectKeyIdentifierExt Plug-in Module

The `SubjectKeyIdentifierExt` plug-in module implements the subject key identifier policy. This policy enables you to configure Certificate Management System to add the *Subject Key Identifier Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) to certificates. The extension is used to identify certificates that contain a particular public key—that is, the extension is used to uniquely identify a certificate from among several that have the same subject name.

Typically, the subject key identifier extension is used in CA certificates as it helps determine which CA key is being certified in a CA certificate. To facilitate chain building, you should consider adding this extension to conforming subordinate CA certificates (subordinate Certificate Managers' CA signing certificates) issued by Certificate Management System. You may also want to consider adding this extension to other or all certificates. For example, if added to end-entity certificates, the 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 that contain a particular public key.

For general guidelines on setting the subject key identifier extension, see “`subjectKeyIdentifier`” on page 360.

The subject key identifier extension policy in Certificate Management System allows setting of the subject key identifier extension as defined in its X.509 definition. It enables you to specify the method for forming the Key Identifier.

By default, the policy supports three types of methods for deriving the Key Identifier; the default methods for forming the Key Identifier are based on PKIX recommendations as defined in section 4.2.1.2. They are as follows:

- 20 byte (160 bit) SHA-1 hash of the BIT STRING of Subject Public Key.
- A type field value of 0100 followed by 60 least significant bits of the SHA-1 hash of the Subject Public Key.
- 20 byte (160 bit) SHA-1 hash of the Subject Public Key Info. This is how Netscape Communicator generates a Key Identifier (but is not necessary to be compatible with the Communicator).

You can also customize the method for deriving the Key Identifier by subclassing the policy and overriding the following method:

```
formKeyIdentifier(X509CertInfo certInfo, IRequest req)
```

For details, check the CMS SDK installed at this location:

```
<server_root>/cms_sdk/cms_jdk/javadocs
```

You may also want to check the CMS samples installed here:

```
<server_root>/cms_sdk/cms_jdk/samples/policies
```

If enabled, the policy adds a Subject Key Identifier Extension to an enrollment request if the extension does not already exist. If the extension exists in the request, for example from a CRMF request, the policy replaces the extension. In case of manual enrollments, after an agent approves the enrollment request, the policy accepts any Subject Key Identifier Extension that is already there.

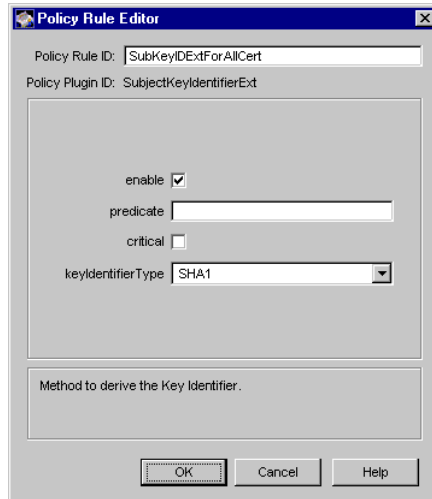
During installation, Certificate Management System automatically creates an instance of the subject key identifier extension policy. See “SubjectKeyIdentifierExt Rule” on page 248.

Configuration Parameters of SubjectKeyIdentifierExt

In the CMS configuration file, the SubjectKeyIdentifierExt module is identified as `<subsystem>.Policy.impl.SubjectKeyIdentifierExt.class=com.netscape.certsrv.policy.SubjectKeyIdentifierExt`, where `<subsystem>` is `ca` or `ra` (prefix identifying the subsystem).

In the CMS window, the module is identified as `SubjectKeyIdentifierExt`. Figure 4-29 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 4-29 Parameters defined in the `SubjectKeyIdentifierExt` module



The configuration shown in Figure 4-29 creates a policy rule named `SubKeyIDExtForAllCert`, which enforces a rule that the server should set the subject key identifier extension in all certificates.

Table 4-28 provides details for each of these parameters.

Table 4-28 Description of configuration parameters defined in the `SubjectKeyIdentifierExt` module

Parameter	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule. Uncheck the box to disable the rule (default).</p> <ul style="list-style-type: none"> If you enable the rule and set the remaining parameters correctly, the server adds the subject key identifier extension to certificates specified by the <code>predicate</code> parameter. If you disable the rule, the server does not add the extension to certificates; it ignores the values in the remaining fields.

Table 4-28 Description of configuration parameters defined in the SubjectKeyIdentifierExt module

Parameter	Description
<code>predicate</code>	Specifies the predicate expression for this rule. If you want this rule to be applied to all certificate requests, leave the field blank (default). To form a predicate expression, see section “Using Predicates in Policy Rules” in Chapter 18, “Setting Up Policies” of <i>CMS Installation and Setup Guide</i> . Example: <code>HTTP_PARAMS.certType==ca</code>
<code>critical</code>	Specifies whether the extension should be marked critical or noncritical in certificates specified by the <code>predicate</code> parameter. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).
<code>KeyIdentifierType</code>	Specifies the method for deriving Key Identifier. Permissible values: <code>SHA1</code> , <code>TypeField</code> , or <code>SpkiSHA1</code> . <ul style="list-style-type: none"> <code>SHA1</code> specifies that the key identifier must be derived as a 20 byte (160 bit) SHA-1 hash of the BIT STRING of Subject Public Key (default). <code>TypeField</code> specifies that the key identifier must be derived as a type field value of 0100 followed by 60 least significant bits of the SHA-1 hash of the Subject Public Key. <code>SpkiSHA1</code> specifies that the key identifier must be derived as a 20 byte (160 bit) SHA-1 hash of the Subject Public Key Info. Example: <code>SHA1</code>

SubjectKeyIdentifierExt Rule

The policy rule named `SubjectKeyIdentifierExt` is an instance of the `SubjectKeyIdentifierExt` module. Certificate Management System automatically creates this rule during installation. By default, the rule is configured as follows:

- The rule is enabled.
- The predicate expression is set (`predicate=HTTP_PARAMS.certType==ca`) so that the extension gets added to CA certificates only. (PKIX and Federal PKI standards recommend that CA certificates must have this extension and end-entity certificates should have this extension.)
- The key identifier is a 20 byte (160 bit) SHA-1 hash of the BIT STRING of Subject Public Key (`KeyIdentifierType=SHA1`).

For details on individual parameters defined in the rule, see Table 4-28 on page 247. It is important that you review this rule and make the appropriate changes required by your PKI setup. For example, if you're planning to issue multiple certificates to an end entity and want to assist applications in identifying the appropriate end-entity certificate, you should consider modifying the predicate expression to add this extension to all end-entity certificates. For instructions, see section "Step 2. Modify Existing Policy Rules" in Chapter 18, "Setting Up Policies" of *CMS Installation and Setup Guide*. For instructions on adding additional instances, see section "Step 4. Add New Policy Rules" in the same chapter.

Mapper Plug-in Modules

You can configure a Certificate Manager to publish certificates to an LDAP directory or flat file, and to publish CRLs to a directory, online validation authority, or flat file. If you configure the Certificate Manager to publish to any of these repositories, when the Certificate Manager is requested to issue a certificate or to update certificate information, it automatically updates the corresponding entry in the configured repository with relevant information. Similarly, when a certificate is revoked, the Certificate Manager automatically updates the configured repository with relevant CRL information. To locate the correct entry in the repository, the Certificate Manager relies on object-mapping rules and to update the located entry with relevant information, the Certificate Manager relies on object-publishing rules.

To enable you to construct object-mapping rules, the Certificate Manager provides a set of mapper plug-in modules. These modules are implemented as Java classes and are registered with the Certificate Manager's publishing framework.

This chapter explains the mapper modules that are installed with a Certificate Manager—it lists and briefly describes the modules and then explains each one in detail.

The chapter has the following sections:

- Overview of Mapper Modules (page 252)
- LdapCaSimpleMap Plug-in Module (page 255)
- LdapDNCompsMap Plug-in Module (page 259)
- LdapDNExactMap Plug-in Module (page 264)
- LdapSimpleMap Plug-in Module (page 265)
- LdapSubjAttrMap Plug-in Module (page 268)

Overview of Mapper Modules

If you configure a Certificate Manager to publish to a directory, whenever the server issues a certificate or updates a certificate or CRL, it needs to locate the entry in the directory in order to update it. For example, to find the correct directory entry to update, the Certificate Manager needs to present Directory Server with search criteria (so that it can initiate an LDAP search operation); the Certificate Manager considers the search successful only if Directory Server returns a single LDAP entry that exactly matches the search criteria.

The Certificate Manager uses object-mapping rules to find the directory entry that needs to be updated. When configuring a Certificate Manager for publishing certificates and CRLs, you define mapping rules that help the server to construct appropriate search criteria that find the entry that needs to be updated.

Mapper modules help you configure the Certificate Manager to use specific rules to map or locate a specific entry, such as a CA's entry or an end-entity's entry, in a specified directory; once the correct entry is located, the server publishes the certificate or CRL to the correct attribute in the entry using a publisher rule, as explained in Chapter 6, "Publisher Plug-in Modules".

By default, the Certificate Manager provides a set of mapper plug-in modules for mapping the CA certificate, end-entity certificates, and CRLs to the appropriate entries in an LDAP directory; because it's not required to map entries in a flat file and online validation authority, no mapper modules are provided for mapping objects in a flat file or an online validation authority.

Plug-in modules are implemented as Java classes and are registered in the CMS publishing framework. The Mapper Plugin Registration tab of the CMS window (Figure 5-1) lists all the modules and the corresponding classes that are registered by default with a Certificate Manager.

Note that the name of the Java class for a mapper plug-in module is in this format:

```
com.netscape.certsrv.ldap.<plugin_name>
```

where <plugin_name> is the name of a plug-in module. For example, the Java class for the `LdapCaSimpleMap` module would be:

```
com.netscape.certsrv.ldap.LdapCaSimpleMap
```

Figure 5-1 Default mapper modules registered with a Certificate Manager

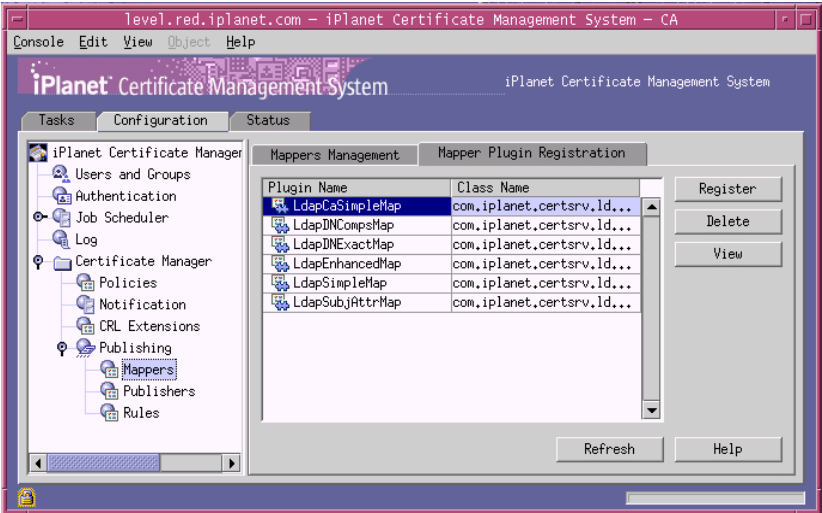


Table 5-1 lists the mapper modules provided for the Certificate Manager.

Table 5-1 Default mapper plug-in modules for mapping certificates and CRLs

Plug-in module name	Function
LdapCaSimpleMap	Maps the CA certificate to the CA's directory entry by formulating the entry's DN from components specified in the certificate's issuer name and attribute variable assertion (AVA) constants. Optionally, the plugin can also create an entry for the CA in the directory. For details, see "LdapCaSimpleMap Plug-in Module" on page 255.
LdapDNCompsMap	Maps a certificate to a directory entry by formulating the entry's 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 "LdapDNCompsMap Plug-in Module" on page 259.
LdapDNExactMap	Maps a certificate to a directory entry by searching for the entry whose DN exactly matches the certificate subject name. For details, see "LdapDNExactMap Plug-in Module" on page 264.
LdapSimpleMap	Maps a certificate to a directory entry by formulating the entry's DN from components specified in the certificate's subject name and attribute variable assertion (AVA) constants. For details, see "LdapSimpleMap Plug-in Module" on page 265.

Table 5-1 Default mapper plug-in modules for mapping certificates and CRLs *(Continued)*

Plug-in module name	Function
LdapSubjAttrMap	Maps a certificate to a directory entry by searching for the entry that contains the LDAP attribute named <code>certSubjNameAttr</code> whose value exactly matches the certificate subject name. For details, see “LdapSubjAttrMap Plug-in Module” on page 268.

After you take a look at the default mapper modules, if you determine that they do not meet your requirements entirely, you can develop a custom mapper module by implementing the following Java interface:

```
com.netscape.certsrv.ldappublish.ILdapMapper
```

For more information about this interface, check the CMS software development kit (SDK) installed at this location:

```
<server_root>/cms_sdk/cms_jdk
```

Be sure to take a look at the samples available at this location:

```
<server_root>/cms_sdk/cms_jdk/samples/mappers
```

When developing a custom mapper module, you may want to intercept LDAP error 52 and reword it so that the correct error message gets logged. To give you an example, if the publishing directory has been stopped, the server logs the following message in its error and system logs:

```
Error publishing CRL MasterCRL: Cannot find a match in the LDAP
server for certificate. netscape.ldap.LDAPException: unable to
establish connection (52); DSA is unavailable.
```

Notice that the error message incorrectly says DSA is unavailable instead of Directory Server is unavailable.

For instructions on how to configure a Certificate Manager to use a mapper module, see section “Configuring a Certificate Manager to Publish Certificates and CRLs” in Chapter 19, “Setting Up LDAP Publishing” of *CMS Installation and Setup Guide*.

LdapCaSimpleMap Plug-in Module

The `LdapCaSimpleMap` plug-in module implements the CA certificate mapper. This mapper enables you to configure a Certificate Manager to automatically create an entry for the CA in an LDAP directory and then map the CA's certificate to the directory entry by formulating the entry's DN from components specified in the certificate request, certificate subject name, certificate extension, and attribute variable assertion (AVA) constants. For more information on AVAs, check the directory documentation.

The CA certificate mapper allows you to specify whether to create an entry for the CA or to just map the certificate to an existing entry, or to do both. For example, you can choose to manually create an entry for the CA in the directory and then configure the CA certificate mapper to just locate the entry by using attributes from the issuer name in the CA's signing certificate and AVA constants.

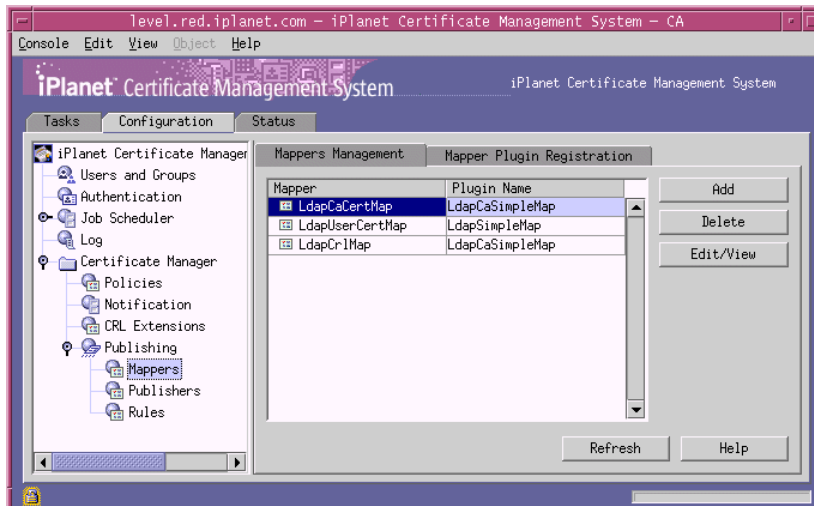
Note that if you already have one CA entry created in the publishing directory and if you change the value assigned to the `dnPattern` parameter of this mapper to something different, but with the same UID and O attributes, the mapper will fail to create the second CA entry. For example, if the directory already has a CA entry with `UID=CA,OU=Marketing,O=Siroe.com` and if you configure the mapper to create another CA entry with `UID=CA,OU=Engineering,O=Siroe.com`, the operation will fail.

The reason for the failure may be because you are using a directory (for example, the configuration directory) that has the `uid uniqueness` plug-in set to a specific base DN in the `slapd.ldbm.conf` file. This setting prevents the directory from having two entries with the same UID under that base DN. For example, it prevents the directory from having two entries under `O=Siroe.com` with the same UID, CA.

If the mapper fails to create a second CA entry, be sure to check the base DN that the `uid uniqueness` plug-in is set to (in the `slapd.ldbm.conf` file) and also check if an entry with the same UID already exists in the directory. If it's true, adjust the mapper setting, remove the old CA entry, comment out the plug-in, or create the entry manually using the CMS window.

During installation, the Certificate Manager automatically creates two instances (called mappers) of the CA certificate mapper module (see Figure 5-2). The mappers are named as follows:

- `LdapCrlMap` for CRLs (see “LdapCrlMap Mapper” on page 259)
- `LdapCaCertMap` for CA certificates (see “LdapCaCertMap Mapper” on page 258)

Figure 5-2 Default mappers created during installation

It is important that you review and customize these mappers. For instructions on modifying mappers or creating new mappers, section “Configuring a Certificate Manager to Publish Certificates and CRLs” in Chapter 19, “Setting Up LDAP Publishing” of *CMS Installation and Setup Guide*.

Configuration Parameters of LdapCaSimpleMap

In the CMS configuration file, the LdapCaSimpleMap module is identified as `ca.publish.mapper.impl.LdapCaSimpleMap.class=com.netscape.certsrv.ldap.LdapCaSimpleMap`.

In the CMS window, the module is identified as LdapCaSimpleMap. Figure 5-3 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 5-3 Parameters defined in the LdapCaSimpleMap module

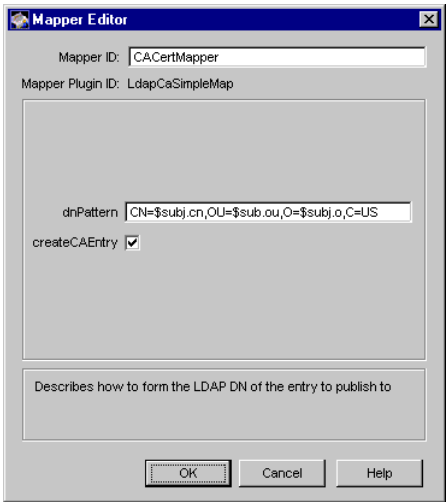


Table 5-2 describes these parameters.

Table 5-2 Description of parameters defined in the LdapCaSimpleMap module

Parameter	Description
createCAEntry	<p>Specifies whether the Certificate Manager should create an entry for the CA in the publishing directory. Check the box if you want the server to create a CA's entry (default). Uncheck the box if you don't want the server to create an entry.</p> <p>If you check the box, the Certificate Manager first attempts to create an entry for the CA in the directory. If the Certificate Manager succeeds in creating the entry, it then attempts to publish the CA's certificate to the entry. Note that the CA's entry DN in the directory will match the pattern you specify in the <code>dnPattern</code> field. For example, if the issuer DN (specified in the CA's signing certificate) is <code>CN=testCA, OU=Research Dept, O=Siroe Corporation, C=US</code>, and the <code>dnPattern</code> is set to <code>CN=\$subj.cn,OU=\$subj.ou,O=\$subj.o,C=US</code>, the Certificate Manager creates an entry with <code>CN=testCA, OU=Research Dept, O=Siroe Corporation, C=US</code> as its DN.</p>

Table 5-2 Description of parameters defined in the LdapCaSimpleMap module *(Continued)*

Parameter	Description
dnPattern	<p>Specifies the DN pattern the Certificate Manager should use to construct the DN in order to search for the CA's entry in the publishing directory. The value of <code>dnPattern</code> can be a list of AVAs separated by commas. An AVA can be a variable, such as <code>CN=\$subj.cn</code>, that the Certificate Manager can derive from the certificate subject name, or a constant, such as <code>O=Siroe Corporation</code>.</p> <p>Note that if your CA certificate does not have the <code>CN</code> component in its subject name, be sure to adjust the CA certificate mapping DN pattern to reflect the DN of the entry in the directory where the CA certificate is to be published. For example, if your CA certificate subject DN is <code>O=Siroe Corporation</code> and the CA's entry in the directory is <code>cn=Certificate Authority, o=Siroe Corporation</code>, the pattern should look like this: <code>cn=Certificate Authority, o=\$subj.o</code></p> <p>(This rule applies to other mappers as well.)</p> <p>Permissible values: A valid pattern that will enable the Certificate Manager to construct the DN for the CA's entry.</p> <p>Example 1: <code>uid=CertMgr, o=Siroe Corporation</code></p> <p>Example 2: <code>CN=\$subj.cn,OU=\$subj.ou,O=\$subj.o,C=US</code></p> <p>Example 3: <code>uid=\$req.HTTP_PARAMS.uid, E=\$ext.SubjectAlternativeName.RFC822Name,ou=\$subj.ou</code></p> <p>In the above examples, <code>\$req</code> means take the attribute from the certificate request, <code>\$subj</code> means take the attribute from the certificate subject name, and <code>\$ext</code> means take the attribute from the certificate extension.</p>

LdapCaCertMap Mapper

The mapper named `LdapCaCertMap` is an instance of the `LdapCaSimpleMap` module. The Certificate Manager automatically creates this mapper during installation.

You can use this mapper for creating an entry for the CA in the directory and for mapping the CA certificate to the CA's entry in the directory.

By default, the mapper is configured to create an entry for the CA in the directory and the default DN pattern for locating the CA's entry is as follows:

`UID=$subj.cn,OU=people,O=$subj.o`

LdapCrlMap Mapper

The mapper named `LdapCrlMap` is an instance of the `LdapCaSimpleMap` module. The Certificate Manager automatically creates this mapper during installation.

You can use this mapper for creating an entry for the CA in the directory and for mapping the CRL to the CA's entry in the directory.

By default, the mapper is configured to create an entry for the CA in the directory and the default DN pattern for locating the CA's entry is as follows:

```
UID=$subj.cn,OU=people,O=$subj.o
```

LdapDNCompsMap Plug-in Module

The `LdapDNCompsMap` plug-in module implements the DN components mapper. This mapper enables you to configure a Certificate Manager to map a certificate to an LDAP directory entry by constructing the entry's distinguished name from components (such as `CN`, `OU`, `O`, and `C`) specified in the certificate subject name, and then using it as the search DN to locate the entry in the directory. You can use this mapper to locate the following:

- The CA's entry in the directory for publishing the CA certificate and the CRL.
- End-entity entries in the directory for publishing end-entity certificates.

The mapper requires you to specify values for three parameters, `filterComps`, `dnComps`, and `baseDN`, which are explained in Table 5-3. In general, the mapper takes DN components to build the search DN. The mapper 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.

Note that both `DNComps` and `filterComps` parameters accept valid DN components or attributes separated by commas. The parameters don't accept multiple entries of an attribute; for example, you can set `filterComps` to `CN,OU`, but not to `CN,OU2,OU1`. If there's a need for you to support such a filter, for example, if your directory entries contain multiple `OUS` and you want to use multiple `OUS` in your `filterComps` for filtering entries, you can modify the source code for the `LdapDNCompsMap` module. The java class for the module is in this directory: `<server_root>/cms_sdk/cms_jdk/samples/mappers`

The discussion below explains how mapping by DN components works. It is recommended that you read this before configuring a Certificate Manager to use this mapper.

Subject names in certificates are in distinguished-name format. A *distinguished name* (DN) uniquely identifies an entry in an LDAP directory. The DN consists of components that help identify the entry; for details, see Appendix , “Distinguished Names.”

The following components are commonly used in DNs:

- `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 Siroe Corporation, which is located in Mountain View in the state of California, United States:

```
CN=Jane Doe, E=jdoe@siroe.com, OU=Sales, O=Siroe Corporation,
L=Mountain View, ST=California, C=US
```

The Certificate Manager 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=Siroe Corporation, L=Mountain View,
ST=California, C=US
```

The Certificate Manager can use some or all of these components (`CN`, `OU`, `O`, `L`, `ST`, and `C`) to build a DN for searching the directory. When creating a mapper rule, you can specify the 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 5-3 on page 263.

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, the Certificate Manager 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=Sales, O=Siroe Corporation, C=US
```

Note the following:

- A subject name does not need to have all of the components that you specify for the `dnComps` parameter. The server ignores any components that are not part of the subject name (such as `L`, `ST`, and `E` in this example).
- Unspecified components are not used to build the DN. In the example, if you did not include the `OU` component, the server would use this DN as the base for searching the directory: `CN=Jane Doe, O=Siroe Corporation, C=US`

In general, for the `dnComps` parameter, you should enter those DN components that the Certificate Manager 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, the Certificate Manager might not get a single, distinct matching entry from the DN. For example, the subject name

```
CN=Jane Doe, OU=Sales, O=Siroe Corporation, C=US
```

might match two users with the name Jane Doe in the directory. If that occurred, the Certificate Manager would need additional criteria to determine which entry corresponds to the subject of the certificate.

To specify the components the Certificate Manager must use to distinguish between different entries in the directory, use the `filterComps` parameter; for details, see Table 5-3 on page 263. 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 DNs 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 DN, you can set up the subject names of certificates to include the `UID` component.

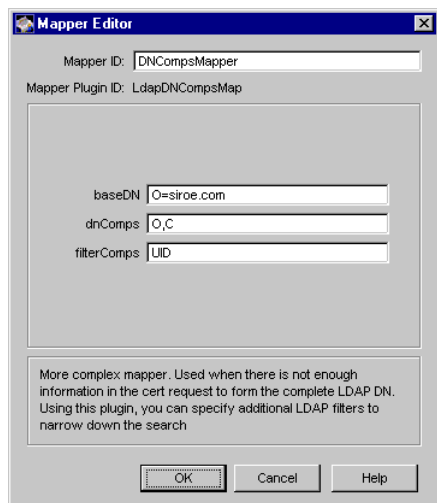
NOTE Generally, the E, L, and ST components are not included in the standard set of certificate request forms provided for end entities. You can add these components to the forms, or you can have the issuing agents insert these components when editing the subject name in the certificate issuance forms.

Configuration Parameters of LdapDNCompsMap

In the configuration file, the `LdapCertCompsMap` module is identified as `ca.publish.mapper.impl.LdapDNCompsMap.class=com.netscape.certsrv.ldap.LdapCertCompsMap`.

In the CMS window, the module is identified as `LdapDNCompsMap`. Figure 5-4 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 5-4 Parameters defined in the LdapDNCompsMap module



With this configuration, a Certificate 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 5-3 describes these parameters.

Table 5-3 Description of parameters defined in the LdapDNCompsMap module

Parameter	Description
<code>baseDN</code>	<p>Specifies the DN to start searching for an entry in the publishing directory. If you leave the <code>dnComps</code> field blank, the server uses the base DN value to start its search in the directory.</p> <p>Permissible values: Alphanumeric string up to 255 characters; see “Base Distinguished Name” on page 315.</p> <p>Example: <code>O=siroe.com</code></p>
<code>dnComps</code>	<p>Specifies where in the publishing directory the Certificate Manager should start searching for an LDAP entry that matches the CA’s or the end entity’s information (that is, the owner of the certificate).</p> <p>The server uses the <code>dnComps</code> values to form an LDAP entry to begin a subtree search. 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=<org></code>, <code>C=<country></code> entry in the directory, where <code><org></code> and <code><country></code> are replaced with values from the DN in the certificate.</p> <p>If you leave the <code>dnComps</code> field empty, the server checks the <code>baseDN</code> field and searches the directory tree specified by that DN for entries matching the filter specified by <code>filterComps</code> parameter values.</p> <p>Permissible values: Valid DN components or attributes separated by commas.</p> <p>Example: <code>O,C</code></p>

Table 5-3 Description of parameters defined in the LdapDNCompsMap module *(Continued)*

Parameter	Description
filterComps	<p>Specifies components the Certificate Manager should use to filter entries from the search result. 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 certificate subject name; 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 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 information gathered from the 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>Permissible values: Valid directory attributes (in the certificate DN) separated by commas. 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>Example: <code>UID</code></p>

LdapDNExactMap Plug-in Module

The `LdapDNExactMap` plug-in module implements the subject name mapper. This mapper enables you to configure a Certificate Manager to map a certificate to an LDAP directory entry by searching for the LDAP entry DN that matches the certificate subject name. Note that to be able to use this mapper, each certificate subject name must exactly match a DN in a directory entry. For example, assume the certificate subject name is this: `UID=jdoe, O=Siroe Corporation, C=US`

When searching the directory for the entry, the Certificate Manager only searches for an entry whose DN is this: `UID=jdoe, O=Siroe Corporation, C=US`

If no matching entries are found, the server returns an error and does not publish the certificate.

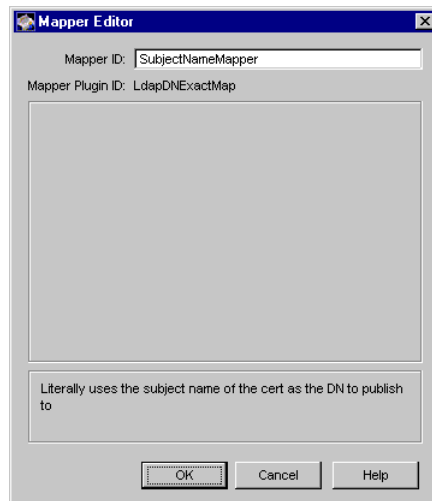
This mapper does not require you to specify any values for any parameters because it obtains all values from the certificate (Figure 5-5).

Configuration Parameters of LdapDNExactMap

In the configuration file, the `LdapDNExactMap` module is identified as `ca.publish.mapper.impl.LdapDNExactMap.class=com.netscape.certsrv.ldap.LdapCertExactMap`.

In the CMS window, the module is identified as `LdapDNExactMap`. Figure 5-5 shows how the module looks when viewed in the CMS window.

Figure 5-5 The `LdapDNExactMap` module



LdapSimpleMap Plug-in Module

The `LdapSimpleMap` plug-in module implements the simple mapper. This mapper enables you to configure a Certificate Manager to map a certificate to an LDAP directory entry by formulating the entry's DN from components specified in the certificate request, certificate's subject name, certificate extension, and attribute variable assertion (AVA) constants. For more information on AVAs, see the directory documentation.

The simple mapper requires you to specify just one parameter, which is named `dnPattern`. The value of `dnPattern` can be a list of AVAs separated by commas. An AVA can be a variable, such as `UID=$subj.UID`, or a constant, such as `O=Siroe Corporation`. The examples below illustrate how you can use AVAs to form the DN pattern.

Example 1: uid=CertMgr, o=Siroe Corporation

Example 2: CN=\$subj.cn,OU=\$subj.ou,O=\$subj.o,C=US

Example 3: uid=\$req.HTTP_PARAMS.uid,
E=\$ext.SubjectAlternativeName.RFC822Name,ou=\$subj.ou

In the above examples, `$req` means take the attribute from the certificate request, `$subj` means take the attribute from the certificate subject name, and `$ext` means take the attribute from the certificate extension.

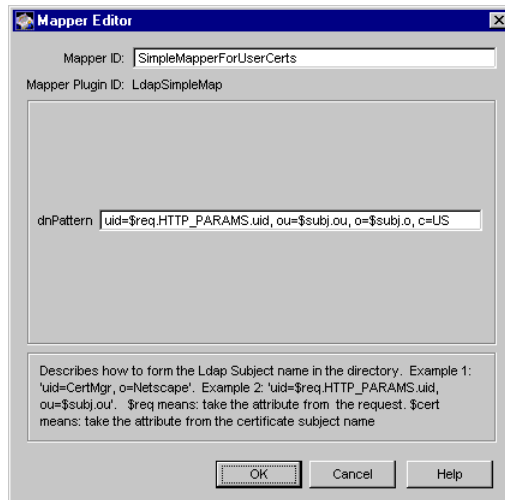
By default, the Certificate Manager uses mapper rules that are based on the simple mapper. During installation, the Certificate Manager automatically creates an instance (called a mapper) of the simple mapper module. The mapper is named `LdapUserCertMap` (see Figure 5-2 on page 256). You can use the default mapper to map various types of end-entity certificates the server will issue to their corresponding directory entries. For details, see “LdapUserCertMap Mapper” on page 267.

It is important that you review and customize this mapper. For instructions on modifying mappers or creating new mappers, section “Configuring a Certificate Manager to Publish Certificates and CRLs” in Chapter 19, “Setting Up LDAP Publishing” of *CMS Installation and Setup Guide*.

Configuration Parameters of LdapSimpleMap

In the CMS configuration file, the `LdapSimpleMap` module is identified as `ca.publish.mapper.impl.LdapSimpleMap.class=com.netscape.certsrv.ldap.LdapSimpleMap`.

In the CMS window, the module is identified as `LdapSimpleMap`. Figure 5-6 shows how configurable parameters for the module are displayed in the CMS window.

Figure 5-6 Parameters defined in the LdapSimpleMap module

LdapUserCertMap Mapper

The rule named `LdapUserCertMap` is an instance of the `LdapSimpleMap` module. The Certificate Manager automatically creates this mapper during installation.

You can use this mapper for mapping end-user certificates to users' directory entries. The default DN pattern for locating end-user entries is as follows:

```
UID=$subj.UID, OU=people, O=$subj.o
```

The default pattern indicates that the Certificate Manager should use the `UID` and `o` values from the certificate subject name and a constant `OU=people` to construct the DN pattern in order to search for an entry.

For example, if the certificate subject name is

```
CN=Jane Doe, UID=jdoe, OU=people, O=Siroe Corporation, C=US
```

the Certificate Manager will construct the following DN to search the directory for the entry:

```
UID=jdoe, OU=people, O=Siroe Corporation
```

LdapSubjAttrMap Plug-in Module

The `LdapSubjAttrMap` plug-in module implements the subject attribute mapper. This mapper enables you to configure a Certificate Manager to map a certificate to an LDAP directory entry by using the LDAP attribute named `certSubjectDN`. Note that for you to be able to use this mapper, your directory entries must include the `certSubjectDN` attribute.

This mapper requires you to specify the exact pattern of the subject DN because the Certificate Manager searches the directory for the `certSubjectDN` attribute whose value exactly matches the entire subject DN specified in the mapper configuration. For example, assume the certificate subject name is this:

```
UID=jdoe, O=Siroe Corporation, C=US
```

When searching the directory for the entry, the Certificate Manager first searches for entries that have these attributes in common

```
certSubjectDN=UID=jdoe, O=Siroe Corporation, C=US
```

and then narrows down the search to an entry that has only this:

```
certSubjectDN=UID=jdoe, O=Siroe Corporation, C=US
```

If no matching entries are found, the server returns an error and writes it to the log; see section “Monitoring CMS Logs” in Chapter 23, “Managing CMS Logs” of *CMS Installation and Setup Guide*.

Configuration Parameters of LdapSubjAttrMap

In the configuration file, the `LdapSubjAttrMap` module is identified as `ca.publish.mapper.impl.LdapSubjAttrMap.class=com.netscape.certsrv.ldap.LdapCertSubjMap`.

In the CMS window, the module is identified as `LdapSubjAttrMap`. Figure 5-7 shows how configurable parameters for the module are displayed in the CMS window.

Figure 5-7 Parameters defined in the LdapSubjAttrMap module

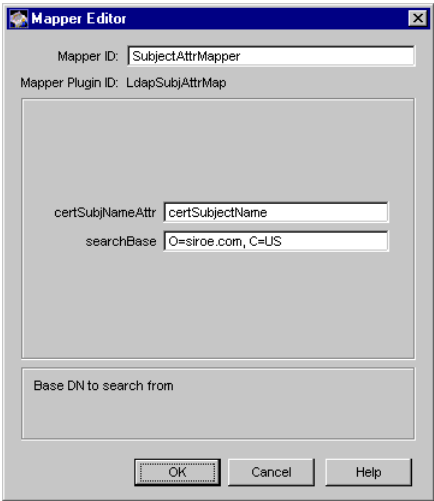


Table 5-4 describes these parameters.

Table 5-4 Description of parameters defined in the LdapSubjAttrMap module

Parameter	Description
certSubjNameAttr	Specifies the name of the LDAP attribute that contains a certificate subject name as its value. Permissible values: Must be certSubjectName. Example: certSubjectName
searchBase	Specifies the base DN for starting the attribute search. Permissible values: A valid DN of an LDAP entry. Example: O=siroe.com, C=US

Publisher Plug-in Modules

You can configure a Certificate Manager to publish certificates to an LDAP directory or flat file, and to publish CRLs to a directory, online validation authority, or flat file. If you configure the Certificate Manager to publish to any of these repositories, when the Certificate Manager is requested to issue a certificate or to update certificate information, it automatically updates the corresponding entry in the configured repository with relevant information. Similarly, when a certificate is revoked, the Certificate Manager automatically updates the configured repository with relevant CRL information. To locate the correct entry in the repository, the Certificate Manager relies on object-mapping rules and to update the located entry with relevant information, the Certificate Manager relies on object-publishing rules.

To enable you to construct object-publishing rules, the Certificate Manager provides a set of publisher plug-in modules. These modules are implemented as Java classes and are registered with the Certificate Manager's publishing framework.

This chapter explains the publisher modules that are installed with a Certificate Manager—it lists and briefly describes the modules and then explains each one in detail. Before reading this chapter, you should have read the previous chapter, Chapter 5, “Mapper Plug-in Modules.”

The chapter has the following sections:

- Overview of Publisher Modules (page 272)
- FileBasedPublisher Plug-in Module (page 274)
- LdapCaCertPublisher Plug-in Module (page 275)
- LdapUserCertPublisher Plug-in Module (page 277)
- LdapCrlPublisher Plug-in Module (page 279)
- OCSPPublisher Plug-in Module (page 281)

Overview of Publisher Modules

Publisher modules help you configure the Certificate Manager to publish a CA certificate, end-entity certificates, or CRLs to the following:

- A mapped entry in the directory (entries are mapped by one of the mapper modules explained in Chapter 5, “Mapper Plug-in Modules.”)
- A particular file
- An online validation authority

By default, the Certificate Manager provides publisher modules for publishing the CA certificate, end-entity certificates, and CRLs. Plug-in modules are implemented as Java classes and are registered in the CMS publishing framework. The Publisher Plugin Registration tab of the CMS window (Figure 6-1) lists all the modules and the corresponding classes that are currently registered with a Certificate Manager.

Figure 6-1 Default publisher modules registered with a Certificate Manager

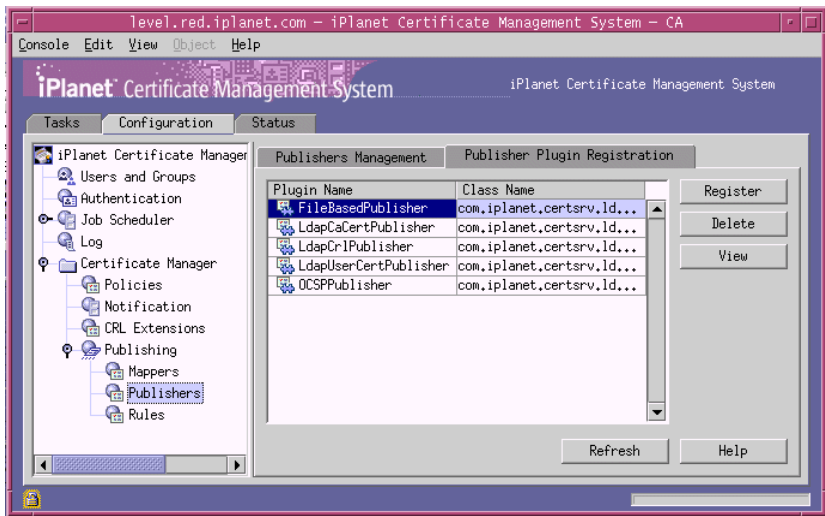


Table 6-1 describes the publisher modules provided for the Certificate Manager. You can use these modules to configure a Certificate Manager to employ specific publishing rules.

Table 6-1 Default publisher plug-in modules for publishing certificates and CRLs

Plug-in module name	Function
<code>FileBasedPublisher</code>	Publishes certificates and CRLs to a flat file (for exporting into other repositories). For details, see “FileBasedPublisher Plug-in Module” on page 274.
<code>LdapCaCertPublisher</code>	Publishes or unpublishes a certificate to the <code>caCertificate;binary</code> attribute of the mapped directory entry as a DER encoded binary blob. Also converts the object class to a <code>certificationAuthority</code> if it’s not one already; similarly, removes the <code>certificationAuthority</code> object class on unpublish if the CA has no other certificates. For details, see “LdapCaCertPublisher Plug-in Module” on page 275.
<code>LdapCrlPublisher</code>	Publishes (replaces) a CRL to the <code>certificateRevocationList;binary</code> attribute of the mapped directory entry as a DER encoded binary blob. The entry should be a <code>certificationAuthority</code> object class. For details, see “LdapCrlPublisher Plug-in Module” on page 279.
<code>LdapUserCertPublisher</code>	Publishes or unpublishes a certificate to the <code>userCertificate;binary</code> attribute of the mapped directory entry as a DER encoded binary blob. For details, see “LdapUserCertPublisher Plug-in Module” on page 277.
<code>OCSPPublisher</code>	Publishes CRLs to a Online Certificate Status Manager. For details, see “OCSPPublisher Plug-in Module” on page 281.

Note that the name of the Java class for a publisher plug-in is in this format:

```
com.netscape.certsrv.ldap.<plugin_name>
```

where `<plugin_name>` is the name of a plug-in module. For example, the Java class for the `FileBasedPublisher` module would be:

```
com.netscape.certsrv.ldap.FileBasedPublisher
```

If you determine that the default publisher modules do not meet your requirements, you can develop a custom publisher class by implementing the following Java interface:

```
com.netscape.certsrv.ldappublish.ILdapPublisher
```

For more information on this interface, check the CMS software development kit (SDK) installed at this location: `<server_root>/cms_sdk/cms_jdk`

Be sure to take a look at the samples available at this location:

```
<server_root>/cms_sdk/cms_jdk/samples/publishers
```

When developing a custom publisher module, you may want to intercept LDAP error 52 and reword it so that the correct error message gets logged. To give you an example, if the publishing directory has been stopped, the server logs the following message in its error and system logs:

```
Error publishing CRL MasterCRL: Cannot find a match in the LDAP
server for certificate. netscape.ldap.LDAPException: unable to
establish connection (52); DSA is unavailable.
```

Notice that the error message incorrectly says DSA is unavailable instead of Directory Server is unavailable.

For instructions on how to configure a Certificate Manager to use a publisher module, see section “Configuring a Certificate Manager to Publish Certificates and CRLs” in Chapter 19, “Setting Up LDAP Publishing” of *CMS Installation and Setup Guide*.

FileBasedPublisher Plug-in Module

The `FileBasedPublisher` plug-in module implements the flat file publisher. This module enables you to configure a Certificate Manager to publish certificates and CRLs to files, which then can be used for importing the certificates and CRLs into any other repository.

By default, the Certificate Manager does not create an instance of the `FileBasedPublisher` module. The instructions covered in Chapter 20, “Publishing Certificates and CRLs to a File” of *CMS Installation and Setup Guide* explain how to create an instance of this module and how to configure a Certificate Manager to publish certificates and CRLs to files.

Configuration Parameters of FileBasedPublisher

In the CMS configuration file, the `FileBasedPublisher` module is identified as `ca.publish.publisher.impl.FileBasedPublisher.class=com.netscape.certsrv.ldap.FileBasedPublisher`.

In the CMS window, the module is identified as `FileBasedPublisher`. Figure 6-2 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 6-2 Configuration parameters defined in the FileBasedPublisher module

The configuration shown in Figure 6-2 creates a publisher named `PublishCertsToFile`, which can publish certificate and CRL files to a directory at `C:\certificates`.

LdapCaCertPublisher Plug-in Module

The `LdapCaCertPublisher` plug-in module implements the CA certificate publisher. This module enables you to configure a Certificate Manager to publish or unpublish a certificate to the `caCertificate;binary` attribute of the mapped directory entry; the mapper must locate the correct entry so the publisher can publish the certificate to the specified attribute. The certificate is published as a DER encoded binary blob.

The module also converts the object class of the CA's entry to a `certificationAuthority` if it's not one already. Similarly, it also removes the `certificationAuthority` object class on unpublish if the CA has no other certificates.

You can use this module for publishing the CA certificate to the LDAP directory only.

During installation, the Certificate Manager automatically creates an instance (called a publisher) of the `LdapCaCertPublisher` module for publishing the CA certificate to the directory. See "LdapCaCertPublisher Publisher" on page 277.

Configuration Parameters of LdapCaCertPublisher

In the CMS configuration file, the `LdapCaCertPublisher` module is identified as `ca.publish.publisher.impl.LdapCaCertPublisher.class=com.netscape.certsrv.ldap.LdapCaCertPublisher`.

In the CMS window, the module is identified as `LdapCaCertPublisher`. Figure 6-3 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 6-3 Parameters defined in the LdapCaCertPublisher module

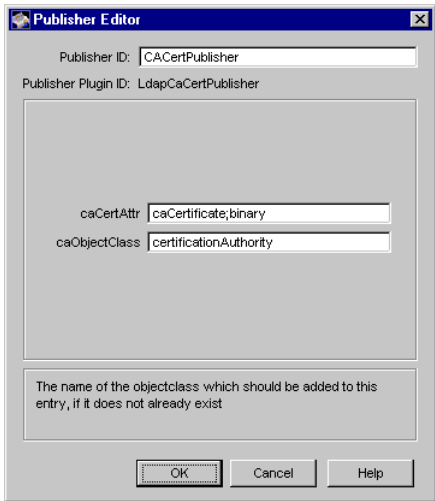


Table 6-2 describes these parameters.

Table 6-2 Description of parameters defined in the LdapCaCertPublisher module

Parameter	Description
caCertAttr	Specifies the LDAP directory attribute to publish the CA certificate. Permissible values: Must be <code>caCertificate;binary</code> . Example: <code>caCertificate;binary</code>

Table 6-2 Description of parameters defined in the LdapCaCertPublisher module (*Continued*)

Parameter	Description
caObjectClass	Specifies the object class for the CA's entry in the directory. Permissible values: Must be <code>certificationAuthority</code> . Example: <code>certificationAuthority</code>

LdapCaCertPublisher Publisher

The publisher named `LdapCaCertPublisher` is an instance of the `LdapCaCertPublisher` module. The Certificate Manager automatically creates this publisher during installation.

You can use this publisher for publishing the CA certificate to `caCertificate;binary` attribute of the mapped CA's entry in the directory.

LdapUserCertPublisher Plug-in Module

The `LdapUserCertPublisher` plug-in module implements the end-entity certificate publisher. This module enables you to configure a Certificate Manager to publish or unpublish a certificate to the `userCertificate;binary` attribute of the mapped directory entry; the mapper must locate the correct entry so the publisher can publish the certificate to the specified attribute. The certificate is published as a DER encoded binary blob.

You can use this module to publish any end-entity certificate to an LDAP directory. Types of end-entity certificates include SSL client, S/MIME, SSL server, object signing, router, and OCSP responder.

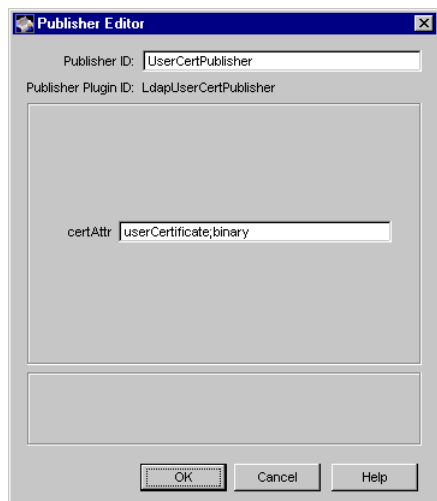
During installation, the Certificate Manager automatically creates an instance (called a publisher) of the `LdapUserCertPublisher` module for publishing end-entity certificates to the directory. See “LdapUserCertPublisher Publisher” on page 279.

Configuration Parameters of LdapUserCertPublisher

In the CMS configuration file, the `LdapUserCertPublisher` module is identified as `ca.publish.publisher.impl.LdapUserCertPublisher.class=com.netscape.certsrv.ldap.LdapUserCertPublisher`.

In the CMS window, the module is identified as `LdapUserCertPublisher`. Figure 6-4 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 6-4 Parameters defined in the LdapUserCertPublisher module



The configuration shown in Figure 6-4 creates a publisher rule named `LdapUserCertPublisher`, which publishes user certificates to the `userCertificate;binary` attribute of the mapped user entries.

Table 6-3 describes the parameters.

Table 6-3 Description of parameters defined in the LdapUserCertPublisher module

Parameter	Description
certAttr	Specifies the directory attribute of the mapped entry to which the Certificate Manager should publish the certificate. Permissible values: Must be <code>userCertificate;binary</code> . Example: <code>userCertificate;binary</code>

LdapUserCertPublisher Publisher

The publisher named `LdapUserCertPublisher` is an instance of the `LdapUserCertPublisher` module. The Certificate Manager automatically creates this publisher during installation.

You can use this publisher to publish an end-entity certificate to the `userCertificate;binary` attribute of the mapped end-entity's entry in the directory.

LdapCrlPublisher Plug-in Module

The `LdapCrlPublisher` plug-in module implements the CRL publisher. This module enables you to configure a Certificate Manager to publish or unpublish the CRL to the `certificateRevocationList;binary` attribute of the mapped directory entry; the configured mapper must locate the CA's entry so that the publisher can publish the CRL to the `certificateRevocationList;binary` attribute. The CRL is published as a DER-encoded binary blob.

The CRL publisher requires you to specify just one parameter named `crlAttr`. The value of this parameter must be `certificateRevocationList;binary`.

During installation, the Certificate Manager automatically creates an instance (called a publisher) of the `LdapCrlPublisher` module for publishing CRLs to the directory. See "LdapCrlPublisher Publisher" on page 281.

Configuration Parameters of LdapCrlPublisher

In the CMS configuration file, the `LdapCrlPublisher` module is identified as `ca.publish.publisher.impl.LdapCrlPublisher.class=com.netscape.certsrv.ldap.LdapCrlPublisher`.

In the CMS window, the module is identified as `LdapCrlPublisher`. Figure 6-5 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 6-5 Parameters defined in the LdapCrlPublisher module

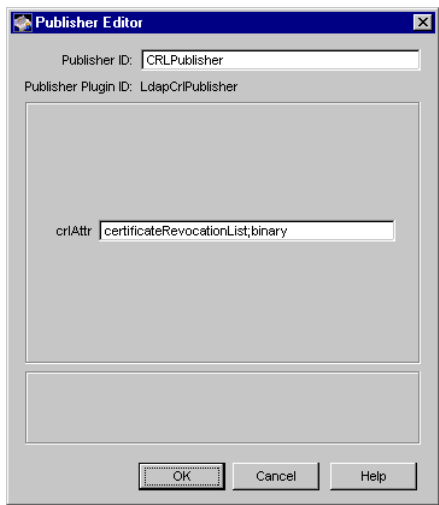


Table 6-4 describes these parameters.

Table 6-4 Description of parameters defined in the LdapCrlPublisher module

Parameter	Description
crlAttr	Specifies the directory attribute of the mapped entry to which the Certificate Manager should publish the certificate. Permissible values: Must be <code>certificateRevocationList;binary</code> . Example: <code>certificateRevocationList;binary</code>

LdapCrlPublisher Publisher

The publisher named `LdapCrlPublisher` is an instance of the `LdapCrlPublisher` module. The Certificate Manager automatically creates this publisher during installation.

You can use this publisher for publishing the CRL to `certificateRevocationList;binary` attribute of the CA's entry in the directory.

OCSPPublisher Plug-in Module

The `OCSPPublisher` plug-in module implements the OCSF publisher. This module enables you to configure a Certificate Manager to publish its CRLs to a Online Certificate Status Manager, the OCSF responder provided by Certificate Management System.

During installation, the Certificate Manager does not create any instances of the `OCSPPublisher` module.

Configuration Parameters of OCSPPublisher

In the CMS configuration file, the `OCSPPublisher` module is identified as `ca.publish.publisher.impl.OCSPPublisher.class=com.netscape.certsrv.ldap.OCSPPublisher`.

In the CMS window, the module is identified as `OCSPPublisher`. Figure 6-6 shows how the configurable parameters for the module are displayed in the CMS window.

Figure 6-6 Parameters defined in the OCSPPublisher module

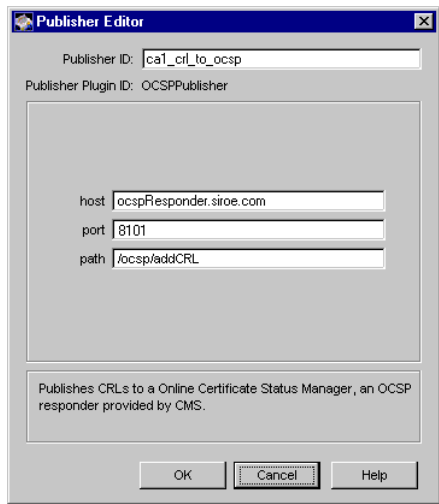


Table 6-5 describes these parameters.

Table 6-5 Description of parameters defined in the OCSPPublisher module

Parameter	Description
host	<p>Specifies the hostname of the Online Certificate Status Manager.</p> <p>Permissible values: Must be the fully-qualified hostname of a Online Certificate Status Manager in this form: <machine>_name>.<your_domain>.com</p> <p>Example: ocspResponder.siroe.com</p>
port	<p>Specifies the port number at which the Online Certificate Status Manager is listening to the Certificate Manager.</p> <p>Permissible values: Must be the Online Certificate Status Manager's agent port number.</p> <p>Example: 8101</p>
path	<p>Specifies the path for publishing the CRL.</p> <p>Permissible values: Must be the default path, /ocsp/addCRL.</p> <p>Example: /ocsp/addCRL</p>

CRL Extension Plug-in Modules

You can configure a Certificate Manager to generate CRLs and publish them to repositories such as an LDAP directory, a flat file, or an OCSP responder which other applications may use for checking the revocation status of a certificate or from which other applications can retrieve the CRL. You can also configure the Certificate Manager to generate and publish CRLs conforming to either X.509 version 1 or X.509 version 2 standards—CRLs compliant to X.509 version 2 standards contain CRL extensions.

To enable you to add these extensions to the CRL it generates, the Certificate Manager provides a set of plug-in modules. These modules are implemented as Java classes and are registered with the Certificate Manager's publishing framework.

This chapter explains plug-in modules that are installed with a Certificate Manager—it lists and briefly describes the modules and then explains each one in detail.

The chapter has the following sections:

- Overview of CRL Extension Modules (page 284)
- AuthorityKeyIdentifier Rule (page 285)
- CRLNumber Rule (page 287)
- CRLReason Rule (page 288)
- HoldInstruction Rule (page 290)
- InvalidityDate Rule (page 291)
- IssuerAlternativeName Rule (page 293)
- IssuingDistributionPoint Rule (page 297)

Overview of CRL Extension Modules

To enable you issue or publish X.509 v2 CRLs (that is, CRLs with extensions), Certificate Management System provides a set of plug-in modules; each module enables you to configure the Certificate Manager to set a particular CRL or CRL-entry extension in CRLs it issues. Plug-in modules are implemented as Java classes and are registered in the CMS publishing framework. The CRL Extensions Management tab of the CMS window (Figure 7-1) lists all the modules that are registered with a Certificate Manager.

When deciding whether to add CRL extensions, keep in mind that not all applications support version 2 CRLs. Among the applications that do support extensions, not all applications will recognize every extension. For general guidelines on using these extensions in CRLs, see Appendix C, “Certificate and CRL Extensions.”

Figure 7-1 Default CRL extension modules registered with a Certificate Manager

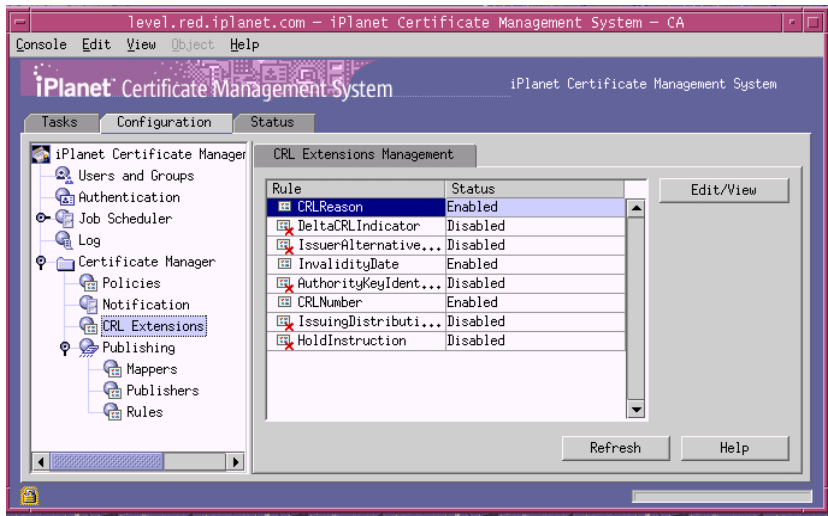


Table 7-1 lists CRL extension modules that are installed with a Certificate Manager. For instructions on how to configure a Certificate Manager to set CRL extensions, see section “Configuring a Certificate Manager to Publish Certificates and CRLs” in Chapter 19, “Setting Up LDAP Publishing” of *CMS Installation and Setup Guide*.

Table 7-1 Default CRL extension modules

Plug-in module name	Function
AuthorityKeyIdentifier	Sets the <i>Authority Key Identifier</i> extension in CRLs. For details, see “AuthorityKeyIdentifier Rule” on page 285.
CRLNumber	Sets the <i>CRL Number</i> extension in CRLs. For details, see “CRLNumber Rule” on page 287.
CRLReason	Sets the <i>Reason Code</i> extension in CRL entries. For details, see “CRLReason Rule” on page 288.
HoldInstruction	Sets the <i>Hold Instruction Code</i> extension in CRL entries. For details, see “HoldInstruction Rule” on page 290.
InvalidityDate	Sets the <i>Invalidity Date</i> extension in CRL entries. For details, see “InvalidityDate Rule” on page 291.
IssuerAlternativeName	Sets the <i>Issuer Alternative Name</i> extension in CRLs. For details, see “IssuerAlternativeName Rule” on page 293.
IssuingDistributionPoint	Sets the <i>Issuing Distribution Point</i> extension in CRLs. For details, see “IssuingDistributionPoint Rule” on page 297.

AuthorityKeyIdentifier Rule

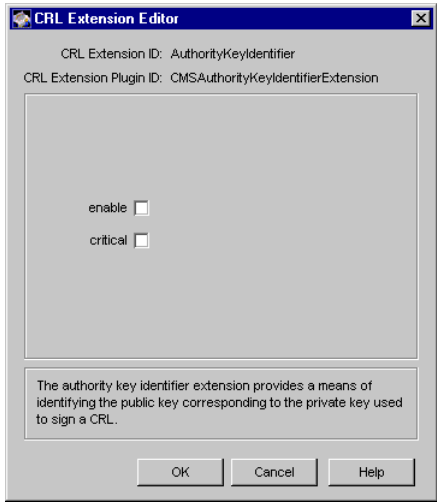
The `AuthorityKeyIdentifier` rule enables you to configure a Certificate Manager to set the *Authority Key Identifier Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) in CRLs. The extension is used to identify the public key that corresponds to the private key used by a CA to sign CRLs.

The PKIX standard recommends that the CA must include this extension in all CRLs it issues. Therefore, you should consider adding this extension to all CRLs issued by the Certificate Manager. The reason for this is that in certain situations, a CA’s public key may change (for example, when the key gets updated) or the CA may have multiple signing keys (either because of multiple concurrent key pairs or because of key changeover). In these cases, the CA ends up with more than one key pair. When verifying a signature on a certificate, other applications need to know which key was used in the signature. The extension, if present in a certificate, enables applications (those that can use the extension) to identify the correct key to use in situations when multiple keys exist; the extension specifies the public key to be used to verify the signature on the CRL.

For general guidelines on setting the authority key identifier extension in CRLs, see “authorityKeyIdentifier” on page 364.

Figure 7-2 shows how configurable parameters for the AuthorityKeyIdentifier rule are displayed in the CMS window.

Figure 7-2 Parameters defined in the AuthorityKeyIdentifier rule



The configuration shown in Figure 7-2 specifies that the server should not set the authority key identifier extension in CRLs.

Table 7-2 describes these parameters.

Table 7-2 Description of parameters defined in the AuthorityKeyIdentifierExt rule

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule. Uncheck the box to disable the rule (default).</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server sets the authority key identifier extension in CRLs.• If you disable the rule, the server does not add the extension to CRLs; it ignores the values in the remaining fields.
critical	<p>Specifies whether the extension should be marked critical or noncritical in CRLs issued by the server. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).</p>

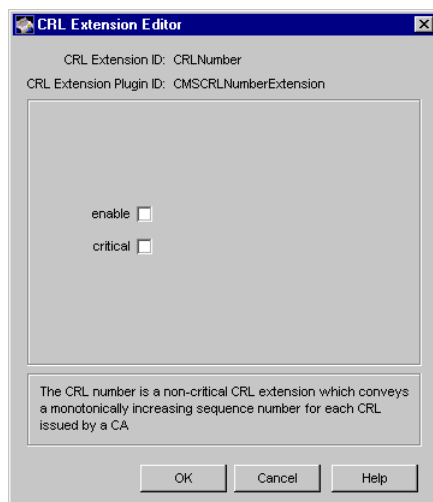
CRLNumber Rule

The `CRLNumber` rule enables you to configure a Certificate Manager to set the *CRL Number Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) in CRLs. This extension specifies a monotonically increasing sequence number for each CRL issued by a CA, allowing CRL users to easily determine when a particular CRL supersedes another CRL.

For general guidelines on setting the CRL number extension in CRLs, see “CRLNumber” on page 365.

Figure 7-3 shows how the configurable parameters for the `CRLNumber` rule are displayed in the CMS window.

Figure 7-3 Parameters defined in the CRLNumber rule



The configuration shown in Figure 7-3 specifies that the server should not set the CRL number extension in CRLs.

Table 7-3 describes these parameters.

Table 7-3 Description of parameters defined in the CRLNumber rule

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule. Uncheck the box to disable the rule (default).</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server sets the CRL number extension in CRLs.• If you disable the rule, the server does not add the extension to CRLs; it ignores the values in the remaining fields.
critical	<p>Specifies whether the extension should be marked critical or noncritical in CRLs issued by the server. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).</p>

CRLReason Rule

The `CRLReason` rule enables you to configure a Certificate Manager to set the *CRL ReasonCode Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) in CRL entries. The extension is used to identify the reason for the revocation of a certificate included in the CRL.

For general guidelines on setting the CRL reason code in CRL entries, see “reasonCode” on page 369.

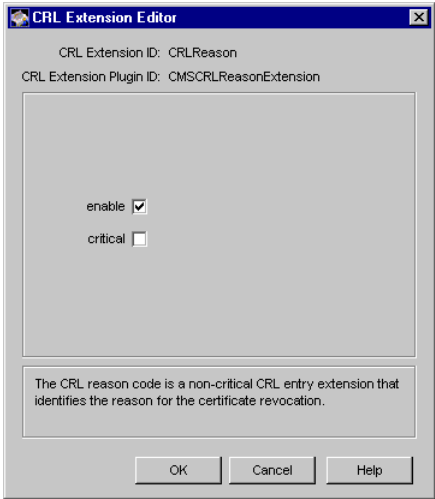
The revocation reasons defined by the standard are listed in Table 7-4.

Table 7-4 Certificate revocation reasons

Code	Reason
0	unspecified
1	keyCompromise
2	cACompromise
3	affiliationChanged
4	superseded
5	cessationOfOperation
6	certificateHold
8	removeFromCRL

Figure 7-4 shows how the configurable parameters for the `CRLReason` rule are displayed in the CMS window.

Figure 7-4 Parameters defined in the CRLReason rule



The configuration shown in Figure 7-4 specifies that the server should set the CRL reason code extension in CRL entries.

Table 7-5 describes these parameters.

Table 7-5 Description of parameters defined in the CRLReason rule

Parameter	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule (default). Uncheck the box to disable the rule.</p> <ul style="list-style-type: none"> • If you enable the rule and set the remaining parameters correctly, the server sets the CRL number extension in CRLs. • If you disable the rule, the server does not add the extension to CRLs; it ignores the values in the remaining fields.
<code>critical</code>	<p>Specifies whether the extension should be marked critical or noncritical in CRLs issued by the server. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).</p>

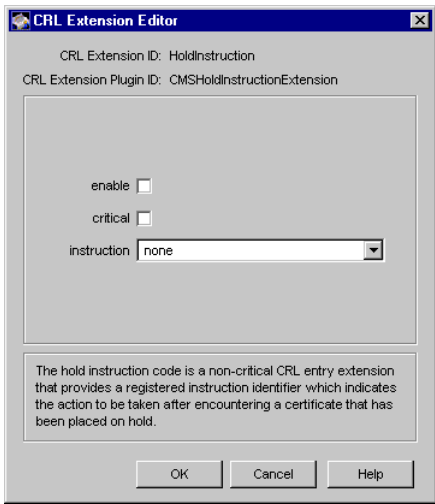
HoldInstruction Rule

The `HoldInstruction` rule enables you to configure a Certificate Manager to set the *CRL Hold Instruction Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) in CRLs. The extension is a non-critical CRL entry extension that is used to specify a registered instruction identifier—the identifier indicates what action the validating application should take when it encounters a certificate that has been placed on hold.

For general guidelines on setting the CRL hold instruction code in CRL entries, see “`holdInstructionCode`” on page 368.

Figure 7-5 shows how the configurable parameters for the `HoldInstruction` rule are displayed in the CMS window.

Figure 7-5 Parameters defined in the `HoldInstruction` rule



The configuration shown in Figure 7-5 specifies that the server should not set the hold instruction extension in CRL entries.

Table 7-6 describes these parameters.

Table 7-6 Description of parameters defined in the HoldInstruction rule

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule. Uncheck the box to disable the rule (default).</p> <ul style="list-style-type: none"> If you enable the rule and set the remaining parameters correctly, the server sets the Hold Instruction extension in CRLs. If you disable the rule, the server does not add the extension to CRLs; it ignores the values in the remaining fields.
critical	<p>Specifies whether the extension should be marked critical or noncritical in CRLs issued by the server. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).</p>
instruction	<p>Specifies the action a validating application must take when it encounters a certificate that has been put on hold.</p> <p>Permissible values: none, callissuer, or reject.</p> <ul style="list-style-type: none"> none specifies that the validating application need not do anything; the PKIX standard says that this is semantically equivalent to the absence of a holdInstructionCode (default). callissuer specifies that the validating application must call the CA that has issued the certificate or reject the certificate. reject specifies that the validating application must reject the certificate on hold. <p>Example: none</p>

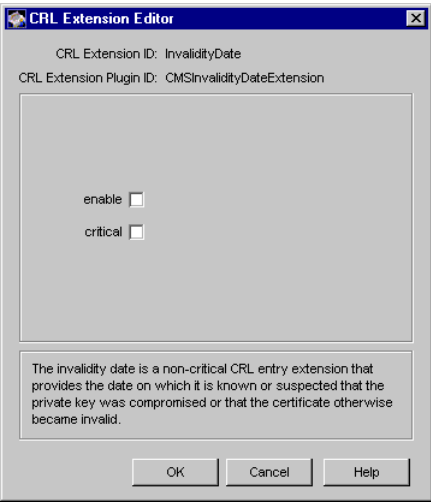
InvalidityDate Rule

The `InvalidityDate` rule enables you to configure a Certificate Manager to set the *Invalidity Date Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) in CRLs. The extension is a non-critical CRL entry extension that is used to specify the date on which it is known or suspected that the private key was compromised or that the certificate otherwise became invalid.

For general guidelines on setting the invalidity date extension in CRL entries, see “invalidityDate” on page 368.

Figure 7-6 shows how the configurable parameters for the `InvalidityDate` rule are displayed in the CMS window.

Figure 7-6 Parameters defined in the `InvalidityDate` rule



The configuration shown in Figure 7-6 specifies that the server should not set the invalidity date extension in CRL entries.

Table 7-7 describes these parameters.

Table 7-7 Description of parameters defined in the `InvalidityDate` rule

Parameter	Description
<code>enable</code>	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule. Uncheck the box to disable the rule (default).</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server sets the Invalidity Date extension in CRLs.• If you disable the rule, the server does not add the extension to CRLs; it ignores the values in the remaining fields.
<code>critical</code>	<p>Specifies whether the extension should be marked critical or noncritical in CRLs issued by the server. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).</p>

IssuerAlternativeName Rule

The `IssuerAlternativeName` rule enables you to configure a Certificate Manager to set the *Issuer Alternative Name Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) in CRLs. This extension enables binding of or associating alternative identities, such as Internet electronic mail address, a DNS name, an IP address, and a uniform resource indicator (URI), with the issuer of the CRL.

The `IssuerAlternativeName` rule enables you to associate the following identities with a CRL issuer, by including them in the extension:

- An `rfc822Name`
- A DNS name
- A directory name
- A uniform resource indicator (URI)
- An IP address
- An object identifier (OID)

For general guidelines on setting the issuer alternative name extension in CRLs, see “`issuerAltName`” on page 366.

Figure 7-7 shows how configurable parameters for the `IssuerAlternativeName` rule are displayed in the CMS window.

Figure 7-7 Parameters defined in the `IssuerAlternativeName` rule

The screenshot shows a dialog box titled "CRL Extension Editor". It contains the following fields and controls:

- CRL Extension ID:** IssuerAlternativeName
- CRL Extension Plugin ID:** CMSIssuerAlternativeNameExtension
- enable:** A checked checkbox.
- critical:** An unchecked checkbox.
- numNames:** A text field containing the value "2".
- nameType0:** A dropdown menu showing "rfc822Name".
- name0:** A text field containing "testCA@sirae.com".
- nameType1:** A dropdown menu showing "dNSName".
- name1:** A text field containing "testCA.sirae.com".
- Check to enable Issuer Alternative Name CRL extension:** A checkbox that is currently unchecked.
- Buttons:** "OK", "Cancel", and "Help" buttons at the bottom.

The configuration shown in Figure 7-7 specifies that the server should not set the issuing point extension in CRLs.

Table 7-8 describes these parameters.

Table 7-8 Description of parameters defined in the IssuerAlternativeName rule

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule. Uncheck the box to disable the rule (default).</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server sets the issuer alternative name extension in CRLs.• If you disable the rule, the server does not add the extension to CRLs; it ignores the values in the remaining fields.
critical	<p>Specifies whether the extension should be marked critical or noncritical in CRLs issued by the server. Check the box if you want the server to mark the extension critical. Uncheck the box if you want the server to mark the extension noncritical (default).</p>
numNames	<p>Specifies the total number of alternative names or identities permitted in the extension. Note that each name has a set of configuration parameters—<code>nameType</code> and <code>name</code>—and you must specify appropriate values for each of those parameters; otherwise, the policy rule will return an error.</p> <p>You can change the total number of identities by changing the value specified in this field; there's no restriction on the total number of identities you can include in the extension. Each set of configuration parameters is distinguished by <code><n></code>, which is an integer derived from the value you assign in this field. For example, if you set the <code>numNames</code> parameter to 2, <code><n></code> would be 0 and 1.</p> <p>Permissible values: 0 or <code>n</code>.</p> <ul style="list-style-type: none">• 0 specifies that no identities can be contained in the extension.• <code>n</code> specifies the total number of identities to be included in the extension; it must be an integer greater than zero. The default value is 3. <p>Example: 1</p>

Table 7-8 Description of parameters defined in the IssuerAlternativeName rule *(Continued)*

Parameter	Description
<code>nameType<n></code>	<p>Specifies the general-name type.</p> <p>Permissible values: <code>rfc822Name</code>, <code>directoryName</code>, <code>dNSName</code>, <code>ediPartyName</code>, <code>URL</code>, <code>iPAddress</code>, <code>OID</code>, or <code>otherName</code>.</p> <ul style="list-style-type: none"> • Select <code>rfc822Name</code> if the name is an Internet mail address. • Select <code>directoryName</code> if the name is an X.500 directory name. • Select <code>dNSName</code> if the name is a DNS name. • Select <code>ediPartyName</code> if the name is a EDI party name. • Select <code>URL</code> if the name is a uniform resource identifier (default). • Select <code>iPAddress</code> if the name is an IP address. • Select <code>OID</code> if the name is an object identifier. • Select <code>otherName</code> if the name is in any other name form. <p>Example: <code>URL</code></p>
<code>name<n></code>	<p>Specifies the general-name value.</p> <p>Permissible values: Depends on the name type specified in the <code>nameType<n></code> field.</p> <ul style="list-style-type: none"> • If the type is <code>rfc822Name</code>, the value must be a valid Internet mail address in the <code>local-part@domain</code> format; see the definition of an <code>rfc822Name</code> as defined in RFC 822 (http://www.ietf.org/rfc/rfc0822.txt). You may use upper and lower case letters in the mail address; no significance is attached to the case. For example, <code>testCA@siroe.com</code>. • If the type is <code>directoryName</code>, the value must be a string form of X.500 name, similar to the subject name in a certificate, in the RFC 2253 syntax (see http://www.ietf.org/rfc/rfc2253.txt). Note that RFC 2253 replaces RFC 1779. For example, <code>CN=CACentral,OU=Research Dept,O=Siroe Corp,C=US</code>. • If the type is <code>dNSName</code>, the value must be a valid domain name in the preferred-name syntax as specified in RFC 1034 (http://www.ietf.org/rfc/rfc1034.txt). You may use upper and lower case letters in the domain name; no significance is attached to the case. Do not use the string “ ” for the DNS name. Also don't use the DNS representation for Internet mail addresses; such identities should be encoded as <code>rfc822Name</code>. For example, <code>testCA.siroe.com</code>.

Table 7-8 Description of parameters defined in the IssuerAlternativeName rule *(Continued)*

Parameter	Description
	<ul style="list-style-type: none">• If the type is <code>ediPartyName</code>, the name must be an <code>IA5String</code>. For example, <code>Siroe Corporation</code>.• If the type is <code>URL</code>, the value must be a non-relative universal resource identifier (URI) following the URL syntax and encoding rules specified in RFC 1738 (http://www.ietf.org/rfc/rfc1738.txt). That is, the name must include both a scheme (for example, <code>http</code>) and a fully qualified domain name or IP address of the host. For example, <code>http://testCA.siroe.com</code>.• If the type is <code>ipAddress</code>, the value must be a valid IP address specified in dot-separated numeric component notation. The syntax for specifying the IP address is as follows: For IP version 4 (IPv4), the address should be in the form specified in RFC 791 (http://www.ietf.org/rfc/rfc0791.txt). IPv4 address must be in the <code>n.n.n.n</code> format; for example, <code>128.21.39.40</code>. IPv4 address with netmask must be in the <code>n.n.n.n,m.m.m.m</code> format. For example, <code>128.21.39.40,255.255.255.00</code>. For IP version 6 (IPv6), the address should be in the form described in RFC 1884 (http://www.ietf.org/rfc/rfc1884.txt), with netmask separated by a comma. Examples of IPv6 addresses with no netmask are <code>0:0:0:0:0:0:13.1.68.3</code> and <code>FF01::43</code>. Examples of IPv6 addresses with netmask are <code>0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0</code> and <code>FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000</code>.• If the type is <code>OID</code>, the value must be a unique, valid OID specified in the dot-separated numeric component notation. Although you can invent your own OIDs for the purposes of evaluating and testing this server, in a production environment, you should comply with the ISO rules for defining OIDs and for registering subtrees of IDs. See Appendix B, “Object Identifiers” for information on allocating private OIDs. For example, <code>1.2.3.4.55.6.5.99</code>.• If the type is <code>otherName</code>, the name must be the absolute path to the file that contains the general name in its base-64 encoded format. For example, <code>C:\netscape\server4\extn\ian\othername.txt</code>.

IssuingDistributionPoint Rule

The `IssuingDistributionPoint` rule enables you to configure a Certificate Manager to set the *Issuing Distribution Point Extension* defined in X.509 and PKIX standard RFC 2459 (see <http://www.ietf.org/rfc/rfc2459.txt>) in CRLs. The CRL issuing point extension enables you to specify a pointer to a particular CRL and to include additional information about the CRL at that location—whether it covers revocation of end-entity certificates only, CA certificates only, or revoked certificates that have a limited set of reason codes.

By default, the pointer can be in either of these forms:

- The name of the X.500 directory that stores the CRL
- The URI to the location that contains the CRL

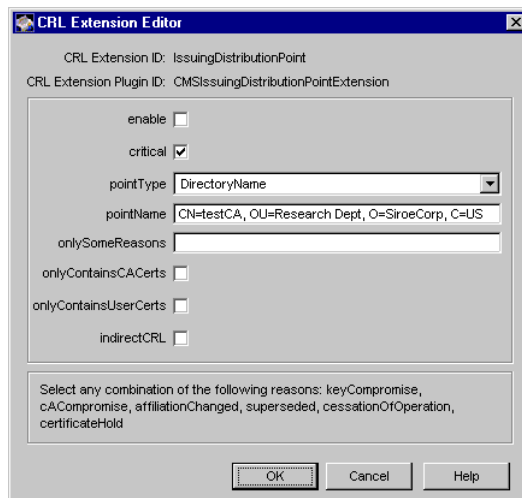
Optionally, each issuing point may contain a set of reason flags, indicating what revocation reasons are covered by the CRL at the specified location. Note that you can modify the rule to support any name form by making the appropriate changes to the sample code provided for this purpose. The sample code is located here:

```
<server_root>/cms_sdk/cms_jdk/samples/CRLs/IssuingDistributionPoint
```

For general guidelines on setting the issuing distribution point extension in CRLs, see “`IssuingDistributionPoint`” on page 366.

Figure 7-8 shows how configurable parameters for the `IssuingDistributionPoint` rule are displayed in the CMS window.

Figure 7-8 Parameters defined in the `IssuingDistributionPoint` rule



The configuration shown in Figure 7-8 specifies that the server should not set the issuing point extension in CRLs.

Table 7-9 describes these parameters.

Table 7-9 Description of parameters defined in the IssuingDistributionPoint rule

Parameter	Description
enable	<p>Specifies whether the rule is enabled or disabled. Check the box to enable the rule. Uncheck the box to disable the rule (default).</p> <ul style="list-style-type: none">• If you enable the rule and set the remaining parameters correctly, the server sets the issuing distribution point extension in CRLs.• If you disable the rule, the server does not add the extension to CRLs; it ignores the values in the remaining fields.
critical	<p>Specifies whether the extension should be marked critical or noncritical in CRLs issued by the server. Check the box if you want the server to mark the extension critical (default). Uncheck the box if you want the server to mark the extension noncritical.</p>
pointType	<p>Specifies the type (for example, URI) of the issuing distribution point.</p> <p>Permissible values: By default, <code>DirectoryName</code> and <code>URI</code>.</p> <ul style="list-style-type: none">• <code>DirectoryName</code> specifies that the type is an X.500 Directory Name (that is, the CRL is stored in an X.500 directory).• <code>URI</code> specifies that the type is a uniform resource indicator (this provides a pointer to the location for the most current CRL issued by this CA). <p>Example: <code>URI</code></p>

Table 7-9 Description of parameters defined in the IssuingDistributionPoint rule *(Continued)*

Parameter	Description
pointName	<p>Specifies the name of the issuing distribution point. The name of the distribution point can be in any of the following formats:</p> <p>Permissible values: Depends on the value specified for the pointType parameter.</p> <ul style="list-style-type: none"> • If the pointType attribute is set to DirectoryName, the name must be an X.500 Name (in RFC1779 syntax). • If the pointType attribute is set to URI, the name must be a URI; the URI must be an absolute pathname and must specify the host. <p>Example:</p> <ul style="list-style-type: none"> • If the name is a URI, it would look similar to this: http://testCA.siroe.com/get/your/crls/here/ • If the name is an X.500 Directory Name, it would look similar to this: CN=CRLCentral,OU=Research Dept,O=Siroe Corp,C=US <p>(Note that the CRL may be stored in the directory entry corresponding to the CRL issuing point, which may be different than the directory entry of the CA.)</p>
onlySomeReasons	<p>Specifies the reason codes associated with the distribution point.</p> <p>Permissible values: A combination of reason codes—unspecified, keyCompromise, cACompromise, affiliationChanged, superseded, cessationOfOperation, certificateHold, and removeFromCRL—separated by commas. Leave field blank if the distribution point contains revoked certificates with all reason codes or if you don't want to set this field (default).</p> <p>Example: unspecified, keyCompromise, cessationOfOperation</p>
onlyContainsCACerts	<p>Specifies whether the distribution point contains only revoked CA certificates. Check the box if the distribution point contains CA certificates only. Uncheck the box if the distribution point contains all types of revoked certificates (default).</p>
onlyContainsUserCerts	<p>Specifies whether the distribution point contains only revoked user certificates. Check the box if the distribution point contains user certificates only. Uncheck the box if the distribution point contains all types of certificates (default).</p>
indirectCRL	<p>Specifies whether the distribution point contains an indirect CRL. Check the box if the distribution point contains an indirect CRL. Uncheck the box if the distribution point doesn't contain an indirect CRL (default).</p>

Log Plug-in Modules

iPlanet Certificate Management Server (CMS) can 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 that the server manages. To monitor these events, you need to capture them in to a repository. For this purpose, Certificate Management System provides plug-in modules. This chapter explains the log modules—it lists and briefly describes the modules and then explains each one in detail.

The chapter has the following sections:

- Overview of Log Modules (page 301)
- file Plug-in Module (page 303)
- NTEventLog Plug-in Module (page 308)

Overview of Log Modules

You can configure a CMS instance to log messages related to specific activities when events relevant to those activities occur. Log messages are event-driven—that is, whenever an event occurs, Certificate Management System generates the message and writes it to the configured repository. Event-driven logging involves a listener class in the CMS instance that registers an interest in an appropriate event such as a failed enrollment request.

Log plug-in modules discussed in this chapter are listeners, which are implemented as Java classes and are registered in the CMS policy framework. The Log Event Listener Plugin Registration tab of the CMS window (Figure 8-1) lists all the modules that are registered with a CMS instance.

Figure 8-1 Default log modules

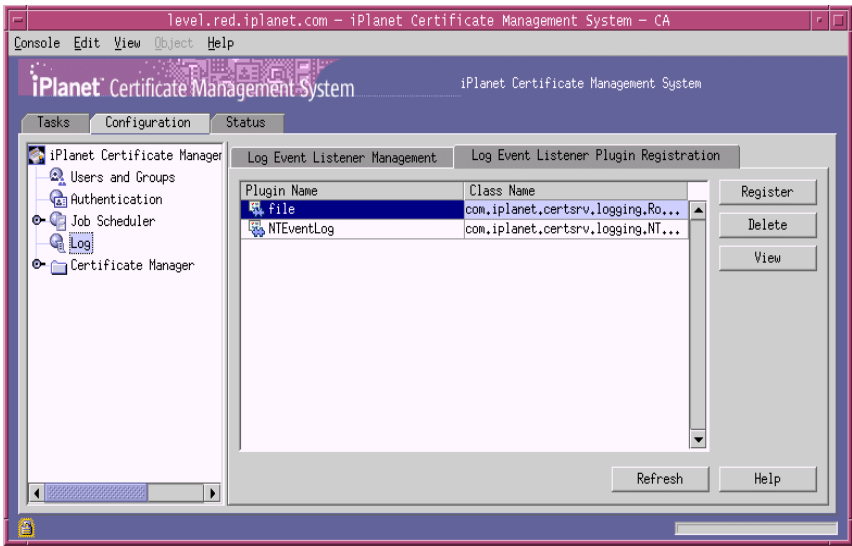


Table 8-1 lists the log modules provided for a CMS instance.

Table 8-1 Log plug-in modules

Plug-in module name	Function
file	Logs messages to a file. For details, see “file Plug-in Module” on page 303.
NTEventLog	Logs messages to Windows NT Event log (when you run a CMS instance on a Windows NT system). For details, see “NTEventLog Plug-in Module” on page 308.

Note that the name of the Java class for a log plug-in is in this format:

```
com.netscape.certsrv.logging.<plugin_name>
```

where <plugin_name> is the name of a plug-in module. For example, the Java class for the NTEventLog module would be:

```
com.netscape.certsrv.logging.NTEventLogs
```

After you take a look at the default log modules, if you determine that they do not meet your requirements entirely, you can develop a custom module. Check the CMS software development kit (SDK) installed at this location:

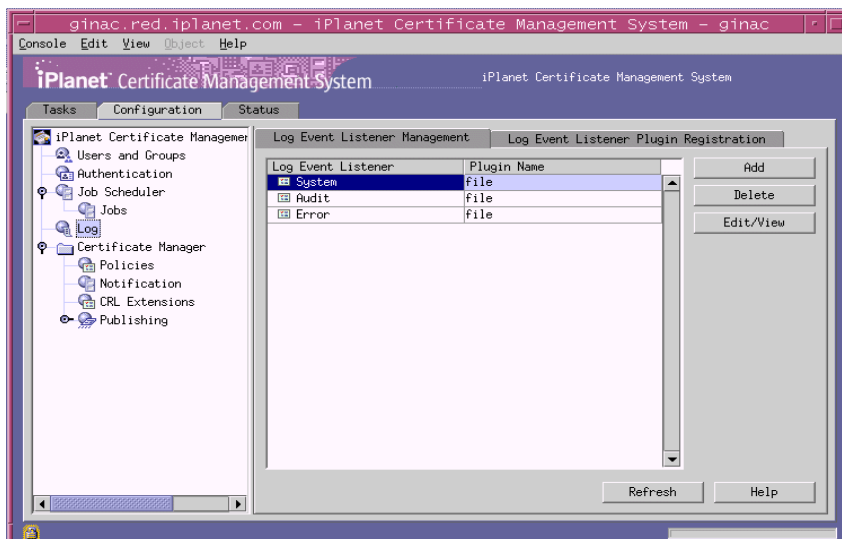
```
<server_root>/cms_sdk/cms_jdk
```

file Plug-in Module

The `file` module enables you to configure Certificate Management System to log audit, error, and system messages to a file. The module also enables you to specify the following:

- Filename
- Log level or message category
- Rollover criteria, which can be based on the size or age of the file
- Expiration time for rotated logs

During installation, Certificate Management System automatically creates three instances of the `file` modules for logging audit, error, and system messages.



The listeners are named as follows:

- Audit (see “Audit Log Event Listener” on page 306)
- Error (see “Error Log Event Listener” on page 307)
- System (see “System Log Event Listener” on page 308)

You need to review these listeners and make the changes appropriate for your PKI setup. For instructions, see “Configuring CMS Logs” in Chapter 23, “Managing CMS Logs” of *CMS Installation and Setup Guide*.

Configuration Parameters of file

In the CMS configuration file, the `file` module is identified as `log.impl.file.class=com.netscape.certsrv.logging.RollingLogFile`.

In the CMS window, the module is identified as `file`. Figure 8-2 shows how configurable parameters for the module are displayed in the CMS window.

Figure 8-2 Parameters defined in the file module

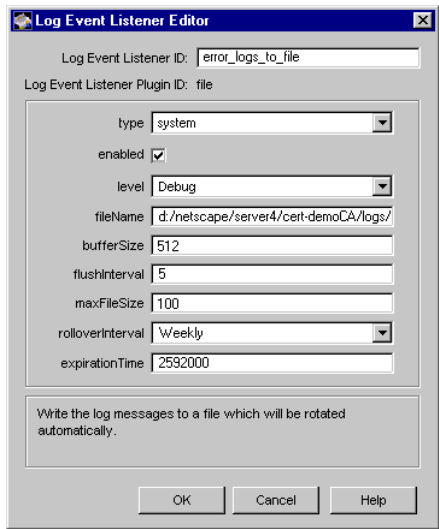


Table 8-2 gives details about each of these parameters and their values.

Table 8-2 Description of parameters defined in the file module

Parameter	Description
type	Specifies the log (or event) type. Permissible values: <code>audit</code> or <code>system</code> . Select <code>audit</code> for Audit logs and <code>system</code> for Error and System logs. The default selection is <code>audit</code> . Example: <code>audit</code>

Table 8-2 Description of parameters defined in the file module *(Continued)*

Parameter	Description
enabled	<p>Specifies whether the listener is enabled to log messages.</p> <ul style="list-style-type: none"> • Check the box if you want the server to log messages of the type specified in the type field. • Leave the box unchecked if you do not want the server to log messages of this type.
level	<p>Specifies a message category that represents the level of logging to filter messages. Log levels are additive. Before selecting a level, be sure to read “Log Levels (Message Categories)” in Chapter 23, “Managing CMS Logs” of <i>CMS Installation and Setup Guide</i>.</p> <p>Permissible values: Debug, Information, Warning, Failure, Misconfiguration, Catastrophe, and Security. By default, the level is set to Information.</p> <p>Example: Debug</p>
fileName	<p>Specifies the file path for the active log file; when the file is rotated, its name will be appended with a timestamp. For details, see “Timing of Log File Rotation” in Chapter 23, “Managing CMS Logs” of <i>CMS Installation and Setup Guide</i>.</p> <p>Permissible values: Absolute path to the file, including the filename.</p> <p>Example: C:\cms\server4\cert-demoCA\logs\audit.log</p>
bufferSize	<p>Specifies the buffer size, in kilobytes (KB), for the active log file. For details, see “Buffered Versus Unbuffered Logging” in Chapter 23, “Managing CMS Logs” of <i>CMS Installation and Setup Guide</i>.</p> <p>Permissible values: As applicable. The default value is 512.</p> <p>Example: 512</p>
flushInterval	<p>Specifies the flush interval, in seconds, for the active log file; when the file reaches the specified interval, the buffer will be flushed to the file. For details, see “Timing of Log File Rotation” in Chapter 23, “Managing CMS Logs” of <i>CMS Installation and Setup Guide</i>.</p> <p>Permissible values: As applicable. The default value is 5.</p> <p>Example: 5</p>

Table 8-2 Description of parameters defined in the file module *(Continued)*

Parameter	Description
maxFileSize	<p>Specifies the file size, in kilobytes (KB), for the active log file; the file will be rotated when its size reaches or exceeds the specified value. For details, see “Timing of Log File Rotation” in Chapter 23, “Managing CMS Logs” of <i>CMS Installation and Setup Guide</i>.</p> <p>Permissible values: As applicable. The default value is 100.</p> <p>Example: 100</p>
rolloverInterval	<p>Specifies the frequency for rotating the active log file; the file will be rotated when its age is equal to or older than this interval. For details, see “Rotation of Log Files” in Chapter 23, “Managing CMS Logs” of <i>CMS Installation and Setup Guide</i>.</p> <p>Permissible values: Hourly, Daily, Weekly, Monthly, and Yearly. The default selection is Hourly.</p> <p>Example: Weekly</p>
expirationTime	<p>Specifies the interval at which the server should delete the rotated log file; the file will be deleted when its age is equal to or older than this interval. By default, the rotated log files are not deleted. For details, see “Timing of Log File Deletion” in Chapter 23, “Managing CMS Logs” of <i>FCMS Installation and Setup Guide</i>.</p> <p>Permissible values: An appropriate value in seconds. For example, if you want the files to be deleted every 30 days, you would type 2592000 (60x60x24x30) seconds.</p> <p>Example: 2592000</p>

Audit Log Event Listener

The event listener named `Audit` is an instance of the `file` module. Certificate Management System automatically creates this listener during installation. By default, the listener is configured as follows:

- The rule is enabled.
- The type is set to log audit messages (`type=audit`).
- The log level for the active log file is set to 1 (`level=Information`).
- Log messages are written to a file named `audit.log`, which is at:
`<server_root>/cert-<instance_id>/logs/`

- The buffer size for the active log file is set to 512 KB (`bufferSize=512`).
- The interval for flushing the buffer to the file is set to 5 seconds (`flushInterval=5`).
- The size limit for the active log file is set to 100 KB (`maxFileSize=100`).
- The rollover interval for the active log file is set to monthly or every 30 days (`rolloverInterval=Monthly`).
- Expiration time for the rotated log files is set to 0 seconds (`expirationTime=0`).

For details on individual parameters defined in the listener, see Table 8-2 on page 304.

Error Log Event Listener

The event listener named `Error` is an instance of the `file` module. Certificate Management System automatically creates this listener during installation. By default, the listener is configured as follows:

- The rule is enabled.
- The type is set to log error messages (`type=system`).
- The log level for the active log file is set to 3 (`level=Failure`).
- Log messages are written to a file named `error.log`, which is at:
`<server_root>/cert-<instance_id>/logs/`
- The buffer size for the active log file is set to 512 KB (`bufferSize=512`).
- The interval for flushing the buffer to the file is set to 5 seconds (`flushInterval=5`).
- The size limit for the active log file is set to 100 KB (`maxFileSize=100`).
- The rollover interval for the active log file is set to monthly or every 30 days (`rolloverInterval=Monthly`).
- Expiration time for the rotated log files is set to 0 seconds (`expirationTime=0`).

For details on individual parameters defined in the listener, see Table 8-2 on page 304.

System Log Event Listener

The event listener named `System` is an instance of the `file` module. Certificate Management System automatically creates this listener during installation. By default, the listener is configured as follows:

- The rule is enabled.
- The type is set to log system messages (`type=system`).
- The log level for the active log file is set to 3 (`level=Failure`).
- Log messages are written to a file named `system.log`, which is at:
`<server_root>/cert-<instance_id>/logs/`
- The buffer size for the active log file is set to 512 KB (`bufferSize=512`).
- The interval for flushing the buffer to the file is set to 5 seconds (`flushInterval=5`).
- The size limit for the active log file is set to 100 KB (`maxFileSize=100`).
- The rollover interval for the active log file is set to monthly or every 30 days (`rolloverInterval=Monthly`).
- Expiration time for the rotated log files is set to 0 seconds (`expirationTime=0`).

For details on individual parameters defined in the listener, see Table 8-2 on page 304.

NTEventLog Plug-in Module

The `NTEventLog` module enables you to configure Certificate Management System to write both audit and system logs to the Event Log of a Windows NT system. If you've installed Certificate Management System on a Windows NT system, the CMS window allows you to turn this feature on or off and to specify the levels for logging.

During installation, Certificate Management System automatically creates two instances or listeners of the `NTEventLog` modules for logging audit and system messages. The listeners are named as follows:

- `NTAudit` (see "NTAudit Event Listener" on page 310)
- `NTSystem` (see "NTSystem Event Listener" on page 310)

Note that by default both the listeners are enabled. You need to review these listeners and make the changes appropriate for your PKI setup. For instructions, see “Configuring CMS Logs” in Chapter 23, “Managing CMS Logs” of *FCMS Installation and Setup Guide*.

Configuration Parameters of NTEventLog

In the configuration file, the NTEventLog module is identified as `log.impl.NTEventLog.class=com.netscape.certsrv.logging.NTEventLog`.

In the CMS window, the module is identified as NTEventLog. Figure 8-3 shows how configurable parameters of the module are displayed in the CMS window.

Figure 8-3 Parameters defined in the NTEventLog module

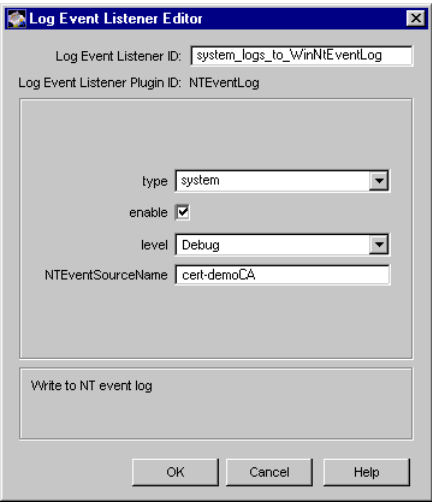


Table 8-3 gives details about each of these parameters and their values.

Table 8-3 Description of parameters defined in the NTEventLog module

Parameter	Description
type	Specifies the log (or event) type. Permissible values: audit or system. Select audit for audit logs and system for error and system logs. Example: system

Table 8-3 Description of parameters defined in the NTEventLog module *(Continued)*

Parameter	Description
enable	<p>Specifies whether the listener is enabled to log messages.</p> <ul style="list-style-type: none"> • Check the box if you want the server to log messages of this type. • Leave the box unchecked if you do not want the server to log messages of this type.
level	<p>Specifies a message category that represents the level of logging to filter messages. For details, see section “Log Levels (Message Categories)” in Chapter 23, “Managing CMS Logs” of <i>CMS Installation and Setup Guide</i>.</p> <p>Permissible values: Debug, Info, Warning, Failure, Misconfiguration, Catastrophe, and Security.</p> <p>Example: Info</p>
NTEventSourceName	Specifies the name of the CMS instance that's logging the messages.

NTAudit Event Listener

The event listener named **NTAudit** is an instance of the NTEventLog module. Certificate Management System automatically creates this listener during installation. By default, the listener is configured as follows:

- The rule is enabled.
- The type is set to log audit messages (type=audit).
- The log level is set to 1 (level=Information).
- The event source identifies the name of the CMS instance that's logging the events.

For details on individual parameters defined in the listener, see Table 8-3 on page 309.

NTSystem Event Listener

The event listener named **NTSystem** is an instance of the NTEventLog module. Certificate Management System automatically creates this listener during installation. By default, the listener is configured as follows:

- The rule is enabled.

- The type is set to log system messages (`type=system`).
- The log level is set to 2 (`level=Warning`).
- The event source identifies the name of the CMS instance that's logging the events.

For details on individual parameters defined in the listener, see Table 8-3 on page 309.

Distinguished Names

This appendix explains what a distinguished name is and how iPlanet Certificate Management Server (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 313)
- DNs in Certificate Management System (page 316)
- Role of Distinguished Names in Certificates (page 323)

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 A-1. For more information on distinguished names, see RFC 2253 (which replaces RFC 1779). You can find RFC 2253 at this URL: <http://www.ietf.org/rfc/rfc2253.txt>

Note that if used in conjunction with an LDAP-compliant directory, Certificate Management System by default recognizes components that are listed in Table A-2.

Table A-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">• CN=Jane Doe• CN=corpDirectory.siroe.com
E (deprecated)	Email address	Identifies the email address of the entry. For example: jdoe@siroe.com The use of this component is discouraged by the PKIX standard; instead, it recommends the use of <i>Subject Alternative Name Extension</i> to associate an email address with a certificate; see “SubjectAltNameExt Plug-in Module” on page 235. The reason for this is because it is usually too hard to have a E in a directory structure; email addresses change too frequently.
OU	Organizational unit	Identifies a unit within the organization. For example: <ul style="list-style-type: none">• OU=Sales• OU=Manufacturing
O	Organization	Identifies the organization in which the entry resides. For example: <ul style="list-style-type: none">• O=Siroe Corporation• O=Public Power & Gas

Table A-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"> • L=Mountain View • L=Pacific Northwest • L=Anoka County
ST	State or province name	Identifies the state or province in which the entry resides. For example: <ul style="list-style-type: none"> • ST=California • ST=British Columbia
C	Country	Identifies the name of the country under which the entry resides. For example: <ul style="list-style-type: none"> • C=US • C=GB
DC	Domain component	Identifies the domain components of a domain. For example, if the domain is siroe.com, the domain components would be: <ul style="list-style-type: none"> • DC=siroe, DC=com

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=Siroe 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=siroe.com for a client, the LDAP search operation initiated by the client examines only the OU=people subtree in the O=siroe.com directory tree.

Typically, an LDAP search consists of the following components:

- The base DN—for example, `O=Siroe, 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=Siroe, 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 Chapter 5, “Mapper Plug-in Modules” and Chapter 6, “Publisher Plug-in Modules.” 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; see section “Configuring a Certificate Manager to Publish Certificates and CRLs” in Chapter 19, “Setting Up LDAP Publishing” of *CMS Installation and Setup Guide*.

DNs in Certificate Management System

In Certificate Management System, the characters allowed in a DN are based on the components (attributes) as defined in the X.509 standard.

Table A-2 lists the attributes supported by default and their character sets. Explanation of the character sets are in Table A-3. The set of attributes is extensible.

Table A-2 Allowed characters for value types

Attribute	Value type	Object identifier
CN	Directory String	2.5.4.3
OU	Directory String	2.5.4.11
O	Directory String	2.5.4.10

Table A-2 Allowed characters for value types *(Continued)*

Attribute	Value type	Object identifier
C	Printable String of length 2	2.5.4.6
L	Directory String	2.5.4.7
ST	Directory String	2.5.4.8
STREET	Directory String	2.5.4.9
TITLE	Directory String	2.5.4.12
UID	Directory String	0.9.2342.19200300.100.1.1
MAIL	IA5String	0.9.2342.19200300.100.1.3
E	IA5String	1.2.840.113549.1.9.1
DC	IA5String	0.9.2342.19200300.100.1.2.25
SERIALNUMBER (for CEP support)	Printable String	2.5.4.5
UNSTRUCTUREDNAME (for CEP support)	IA5String	1.2.840.113549.1.9.2
UNSTRUCTUREDADDRESS (for CEP support)	Printable String	1.2.840.113549.1.9.8

Table A-3 Explanation of character sets for DN's

Value type	Character set allowed
Printable String	A-Z, a-z, 0-9, space \ () + , - . / : = ?
IA5String	Any 7-bit US ASCII character.

Table A-3 Explanation of character sets for DNs *(Continued)*

Value type	Character set allowed
Directory String	<p>Any character in format as specified in <i>Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names</i> (see http://www.ietf.org/rfc/rfc2253.txt). Certificate Management System conforms to all of this standard, including support of using hex numbers to escape characters. The special characters are as follows:</p> <p><code>,</code> <code>=</code> <code>+</code> <code><</code> <code>></code> <code>#</code> <code>;</code></p> <p>They can be escaped by either a backslash (<code>\</code>) before the character or by surrounding the value in double quotes (<code>" "</code>). A few examples are shown below:</p> <p><code>Siroe Corp. \, Ltd.</code> <code>"Siroe Corp. , Ltd"</code> <code>"Siroe Corp. \, Ltd"</code></p> <p>Name <code>\,</code> with <code>\=</code> escaped <code>\+</code> special <code>\<</code> characters <code>\#</code> like <code>\></code> or <code>"\\\"";</code></p> <p><code>"Name , with = special + characters < surrounded > by # quotes; ,+=<>#;"</code></p> <p>Name with escaped characters like return <code>\0D</code> or C with Caron <code>\C4\8D</code> or L<code>\4C</code></p> <p><code> Name with spaces at beginning and end</code></p> <p>For additional more examples, check the standards.</p>

Extending Attribute Support

By default, Certificate Management System supports attribute identified in Table A-2 on page 316. You can extend the list of attributes supported by server. The syntax for adding additional X.500Name attributes (or components) is as follows:

```
X500Name.<NEW_ATTRNAME>.oid=<n.n.n.n>
X500Name.<NEW_ATTRNAME>.class=<string_to_DER_value_converter_class>
```

Note the following:

- Value converter class converts a string to a ASN.1 value.
- It must implement `netscape.security.x509.AVAValueConverter` interface.

The string-to-value converter class can be one of these:

- `netscape.security.x509.PrintableConverter`—converts a string to a Printable String value. The string must have only printable characters.
- `netscape.security.x509.IA5StringConverter`—converts a string to a IA5String value. The string must have only IA5String characters.
- `netscape.security.x509.DirStrConverter`—converts a string to a Directory (v3) String. The string is expected to be in DirectoryString format according to RFC 2253.
- `netscape.security.x509.GenericValueConverter`—converts a string character by character in the following order, from smaller character sets to broadest character set: Printable, IA5String, BMPString, Universal String.

For example:

```
X500Name.MY_ATTR.oid=1.2.3.4.5.6
X500Name.MY_ATTR.class=netscape.security.x509.DirStrConverter
```

Adding New or Proprietary Attributes

To add a new or proprietary attribute that's not supported by Certificate Management System by default:

1. Stop the Certificate Manager.
2. Go to this directory: `<server_root>/cert-<instance_id>/config`
3. Open the configuration file, `CMS.cfg`, in a text editor.
4. Add the new attributes to the configuration file.

For example, if you want to add three proprietary attributes, `MYATTR1` that is a `directoryString`, `MYATTR2` that is a `IA5String`, and `MYATTR3` that is `PrintableStrings`, you would add the following lines at the end of the configuration file:

```
X500Name.attr.MYATTR1.oid=1.2.3.4.5.6
X500Name.attr.MYATTR1.class=netscape.security.x509.
    DirStrConverter
X500Name.attr.MYATTR2.oid=11.22.33.44.55.66
X500Name.attr.MYATTR2.class=netscape.security.x509.
```

```

IA5StringConverter
X500Name.attr.MYATTR3.oid=111.222.333.444.555.666
X500Name.attr.MYATTR3.class=netscape.security.x509.
PrintableConverter

```

5. Save your changes and close the file.
6. Next, add each new attribute or component (for example, MYATTR1, MYATTR2 and MYATTR3) to the enrollment form. For instructions, see “Adding Attributes to an Enrollment Form” on page 320.
7. Restart the Certificate Manager.
8. Reload the enrollment page and verify your changes; the new attributes that you added should now show up in the form.
9. To verify that the new attributes are in effect, request a certificate using the manual enrollment form.

Be sure to enter values for the new attributes (so that you can verify whether they appear in certificate subject names). For example, you can enter the following values for the new attributes and look for them in the subject name:

```

MYATTR1: a_value
MYATTR2: a.Value
MYATTR3: aValue
CN: John Doe
O: Siroe Corp

```

10. Go to the agent interface and approve your request.
11. When you receive the certificate, check the subject name. The certificate should show the new attribute values in the subject name.

Adding Attributes to an Enrollment Form

The steps below explain how to add an attribute (or component) to the Manual enrollment form:

1. Go to this directory: <server_root>/cert-<instance_id>/web/ee
2. Open the ManUserEnroll.html file in a text editor.
3. Find the line with the component name that the new component will follow and copy the table row, using the new component name. For the purposes of this instruction, assume that the new component you want to add is DC and that it follows component OU. Here's how you would add a table row for DC (the lines you need to add are shown in bold):


```

<tr>
  <td valign="TOP">
    <div align="RIGHT">
      <font face="PrimaSans BT, Verdana, Arial, Helvetica,
        sans-serif" size="-1">Organization unit: </font>
    </div>
  </td>
  <td valign="TOP">
    <input type="TEXT" name="OU" size="30"
      onchange="formulatedDN(this.form, this.form.subject)">
  </td>
</tr>

<tr>
  <td valign="TOP">
    <div align="RIGHT">
      <font face="PrimaSans BT, Verdana, Arial, Helvetica,
        sans-serif" size="-1">Domain component: </font>
    </div>
  </td>
  <td valign="TOP">
    <input type="TEXT" name="DC" size="30"
      onchange="formulatedDN(this.form, this.form.subject)">
  </td>
</tr>

```

4. Save your changes and close the file.
5. Go to this directory: <server_root>/cert-<instance_id>/web/ee
6. Open the cms-funcs.js file in a text editor.
7. Find the line with `form.OU != null` (or the component that the new component will follow) and add the `if` block. For example, if the new component is `DC` and comes after `OU`, you need to add the lines shown in bold:

```

if (form.OU != null) {
  if (OU.value != '') {
    if (doubleQuotes(OU.value) == true) {
      alert('Double quotes are not allowed in Org Unit
        field');
      OU.value = '';
      OU.focus();
      return;
    }
    if (distinguishedName.value != '')
      distinguishedName.value += ', ';
  }
}

```

```

        distinguishedName.value += 'OU=' +
        escapeDNComponent(OU.value);
    }
}

if (form.DC != null) {
    if (DC.value != '') {
        if (doubleQuotes(DC.value) == true) {
            alert('Double quotes are not allowed in DC
            field');
            DC.value = '';
            DC.focus();
            return;
        }
        if (distinguishedName.value != '')
            distinguishedName.value += ', ' +
            distinguishedName.value += 'DC=' +
            escapeDNComponent(DC.value);
    }
}

```

8. Save your changes and close the file.
9. Reload the Manual enrollment form in the browser and verify your changes.
10. To verify that the Enroll for a certificate using the new attribute value.

Changing the DER Encoding Order

You can also change the DER-encoding order of a DirectoryString. (The reason for allowing this to be configurable is that different clients support different encodings for historical reasons.)

The syntax for changing the DER-encoding order of a DirectoryString is as follows:

```
X500Name.dirStringEncodingOrder=<encoding_list_separated_by_commas>
```

Possible encoding values are as follows:

- Printable
- IA5String
- UniversalString
- BMPString
- UTF8String

For example, `X500Name.dirEncodingOrder=Printable,BMPString`.

To change the `DirectoryString` encoding:

1. Stop the Certificate Manager.
2. Go to this directory: `<server_root>/cert-<instance_id>/config`
3. Open the configuration file, `CMS.cfg`, in a text editor.
4. Add the encoding order to the configuration file.

For example, if you want to specify two encoding values, `PrintableString` and `UniversalString`, and the encoding order is `PrintableString` first and `UniversalString` next, you would add the following line at the end of the configuration file:

```
X500Name.directoryStringEncodingOrder=PrintableString,
UniversalString
```

5. Save your changes and close the file.
6. To verify that the encoding order are in effect, enroll for a certificate using the manual enrollment form. Use "John_Doe" for CN.
7. Go to the agent interface and approve your request.
8. When you receive the certificate, use the `dumpasn1` tool to examine the encoding of the certificate. For details about the `dumpasn1` tool, see *CMS Command-Line Tools Guide*.

The CN component of the subject name should be encoded as a `UniversalString`.

9. Repeat Steps 6 through 8 above, but use "John Smith" for CN this time.

The CN component of the subject name should be encoded as a `PrintableString`.

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 Chapter 5, "Mapper Plug-in Modules."

DNs in End-Entity Certificates

In end-entity certificates issued by Certificate Management System, DNs are 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=Siroe Corporation, 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=corpDirectory.siroe.com, OU=Human Resources, O=Siroe
Corporation, 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://corpDirectory.siroe.com` and receives a certificate from the server, it expects the **CN** component of the certificate's subject to be `corpDirectory.siroe.com`. If the **CN** component has a different value (for example, `corpDir.siroe.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.

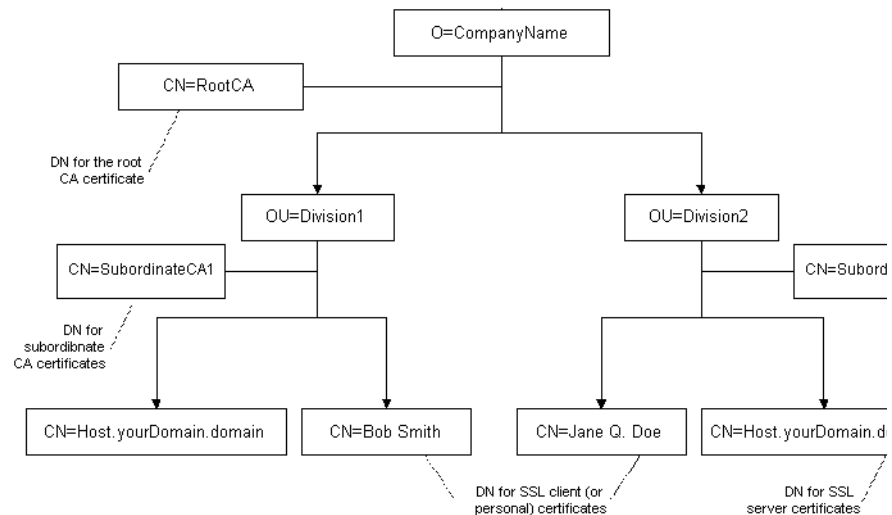
To form this type of distinguished name, use the **CN** component to specify the name of your CA: **CN=<CA_name>, O=<company_name>, C=<country_name>**

For example: CN=Siroe Certificate Authority, O=Siroe Corporation, C=US

Selecting DNs for Certificates

Figure A-1 illustrates the structure of distinguished names you might select for CA certificates, server certificates, and personal certificates.

Figure A-1 Sample directory hierarchy



DN Patterns and Certificate Subject Names

You can configure Certificate Management System to issue certificates with subject names that are formulated from the directory attributes and entry DN. The `dnpattern` configuration variable of the automated-enrollment modules, such as `UidPwdDirAuth` and `UidPwdPinDirAuth`, described in Chapter 1, “Authentication Plug-in Modules” enable you to configure the server to issue certificates with required subject names. Note that `dnpattern` is a string representing a subject name pattern to formulate from the directory attributes and entry DN. If empty or not set, Certificate Management System uses the LDAP entry DN as the certificate subject name.

The `dnpattern` configuration variable supports escaped commas and multiple attribute variable assertions (AVAs) in a RDN. Below is the syntax for the DN pattern followed by examples.

Syntax

```

dnPattern := rdnPattern *["," rdnPattern ]
rdnPattern := avaPattern *["+" avaPattern ]
avaPattern := name "=" value | name "=" "$attr" "." attrName [ "."
    attrNumber ] | name "="
"$dn" "." attrName [ "." attrNumber ] | "$dn" "." "$rdn" "." number

```

Example 1

If the configured DN pattern is

```
E=$attr.mail.1, CN=$attr.cn, OU=$dn.ou.2, O=$dn.o, C=US
```

LDAP entry: dn: UID=jdoe, OU=IS, OU=people, O=siroe.org

LDAP attributes: cn: Jane Doe

LDAP attributes: mail: jdoe@siroe.org

The subject name formulated will be as follows:

```
E=jdoe@siroe.org, CN=Jane Doe, OU=people, O=siroe.org, C=US
```

E= the first 'mail' LDAP attribute value in user's entry.

CN= the (first) 'cn' LDAP attribute value in the user's entry.

OU= the second 'ou' value in the user's entry DN.

O= the (first) 'o' value in the user's entry DN.

C= the string 'US'

Example 2

If the configured DN pattern is

```
E=$attr.mail.1, CN=$attr.cn, OU=$dn.ou.2, O=$dn.o, C=US
```

LDAP entry: dn: UID=jdoe, OU=IS+OU=people, O=siroe.org

LDAP attributes: cn: Jane Doe

LDAP attributes: mail: jdoe@siroe.org

The subject name formulated will be as follows:

```
E=jdoe@siroe.org, CN=Jane Doe, OU=people, O=siroe.org, C=US
```

E= the first 'mail' LDAP attribute value in user's entry.

CN= the (first) 'cn' LDAP attribute value in the user's entry.

OU= the second 'ou' value in the user's entry DN; note the multiple AVAs in a RDN in this example.

O= the (first) 'o' value in the user's entry DN.

C= the string 'US'

Example 3

If the configured DN pattern is

```
CN=$attr.cn, $rdn.2, O=$dn.o, C=US
```

LDAP entry: dn: UID=jdoe, OU=IS+OU=people, O=siroe.org

LDAP attributes: cn: Jane Doe

LDAP attributes: mail: jdoe@siroe.org

The subject name formulated will be as follows:

```
CN=Jane Doe, OU=IS+OU=people, O=siroe.org, C=US
```

CN= the (first) 'cn' LDAP attribute value in the user's entry followed by the second RDN in the user's entry DN.

O= the (first) 'o' value in the user's entry DN.

C= the string 'US'

Example 4

If the configured DN pattern is

```
CN=$attr.cn, OU=$dn.ou.2+OU=$dn.ou.1, O=$dn.o, C=US
```

LDAP entry: dn: UID=jdoe, OU=IS+OU=people, O=siroe, org

LDAP attributes: cn: Jane Doe

LDAP attributes: mail: jdoe@siroe.org

The subject name formulated will be as follows:

```
CN=Jane Doe, OU=people+OU=IS, O="siroe \, org", C=US
```

CN= the (first) 'cn' LDAP attribute value in the user's entry.

OU= the second 'ou' value in the user's entry DN followed by the first 'ou' value in the user's entry; note the multiple AVAs in a RDN in this example.

O= the (first) 'o' value in the user's entry DN.

C= the string 'US'

If an attribute or subject DN component does not exist, the attribute is skipped.

Object Identifiers

iPlanet Certificate Management Server (CMS) comes with a set of extension-specific policy plug-in modules that enable you to add X.509 certificate extensions to the certificates the server issues. Some of the extensions contain fields for specifying object identifiers. This appendix explain what's an object identifier (OID) and the significance of registering it.

The appendix has the following sections:

- What's an Object Identifier? (page 329)
- Registration of Object Identifiers (page 330)

What's an Object Identifier?

An object identifier is a string of numbers identifying a unique object, for example, a certificate extension or a company's certificate practice statement. For general information on OIDs, see the information at this URL:

<http://www.alvestrand.no/objectid/>

OIDs are controlled by the International Standards Organization (ISO) registration authority. In some cases, this authority is delegated by ISO to regional registration authorities. For example, in the United States, the American National Standards Institute (ANSI) manages this registration.

Registration of Object Identifiers

To promote interoperability, the PKIX standard recommends that all objects (such as extensions and policy statements) that appear in certificates that will be used in networks shared by other organizations should be included in the form of OIDs. If you plan to issue certificates that will be used in such networks, you should register your object identifier prefixes with the appropriate registration authority. For example, assume you want to add a custom extension that points to a certificate practice statement (CPS) of your company. To implement this, you need to compose the policy statement you want to include in the extension, define an OID for the policy statement, and configure Certificate Management System with the OID so that it can add that to the certificate it issues.

The use of an OID registered to another organization or the failure to register an OID may carry legal consequences, depending on context. Registration may be subject to fees. For more information, you should contact the appropriate registration authority.

To define or assign OIDs for your objects, you must know your company's *arc*, which is an OID for a private enterprise. If your company doesn't have an *arc*, it needs to get one. This URL contains information on registering for a company *arc*:

<http://www.isi.edu/cgi-bin/iana/enterprise.pl>

To understand why you need to have a company *arc*, check the information at this site:

<http://www.alvestrand.no/objectid/2.16.840.1.113730.1.13.html>

The site contains information on Netscape-defined OID for an extension named Netscape Certificate Comment. Note that the OID assigned to this extension is hierarchical and it includes the Netscape company *arc*, which is 2.16.840.1.113730. Every OID Netscape owns has this prefix.

When determining whether to add custom extension to certificates, keep in mind that if the extension exists in a certificate and if it is marked critical, the application validating the certificate must be able to interpret the extension (including the optional qualifiers, if any), or else it must reject the certificate. Since it's unlikely that all applications will be able to interpret your company's extensions (embedded in the form of OIDs), the PKIX standard recommends that the extension be always marked noncritical. For general guidelines on setting extensions in certificates, see Appendix C, "Certificate and CRL Extensions."

Certificate and CRL Extensions

This appendix explains both the standard certificate extensions defined by X.509 v3 and the extensions defined by Netscape that were used in versions of products released before X.509 v3 was finalized. It also provides recommendations for extensions to use with specific kinds of certificates, including both PKIX Part 1 recommendations and Netscape extensions that must be supported for compatibility with early versions of Netscape products.

This appendix contains the following sections:

- Introduction to Certificate Extensions (page 331)
- Recommendations for Certificate Extension Use (page 335)
- Standard X.509 v3 Certificate Extensions (page 341)
- Introduction to CRL Extensions (page 361)
- Standard X.509 v3 CRL Extensions (page 364)
- Netscape-Defined Certificate Extensions (page 369)
- CA Certificates and Extension Interactions (page 371)

Introduction to Certificate Extensions

An X.509 v3 certificate contains an extensions field that permits any number of additional fields to be added to the certificate. Certificate extensions provide a way of adding information such as alternative subject names and usage restrictions to certificates. Older versions of Netscape browsers and servers that were developed before PKIX part 1 standards were defined require Netscape-specific extensions.

The X.509 v1 certificate specification was originally designed to bind public keys to names in an X.500 directory. As certificates began to be used on the Internet and extranets, and directory lookups could not always be performed, problem areas such as the following emerged that were not foreseen in the original specification:

- **Trust**—The X.500 specification establishes trust by means of a strict directory hierarchy. By contrast, Internet and extranet deployments frequently involve distributed trust models that do not conform to the hierarchical X.500 approach.
- **Certificate use**—Some organizations may wish to restrict the use of certificates for policy reasons. For example, some certificates may be restricted to client authentication only.
- **Multiple certificates**—It's not uncommon for certificate users to possess multiple certificates with identical subject names but different key material. In this case, it's necessary to identify which key and certificate should be used for what purpose.
- **Alternate names**—For some purposes, it is useful to have alternative subject names that are also bound to the public key in the certificate.
- **Additional attributes**—Some organizations may find it convenient to store additional information in certificates, for example for situations in which it's not possible to look up information in a directory.
- **Relationship with CA**—When certificate chaining involves intermediate CAs, it is useful to have information about the relationships among CAs embedded in their certificates.
- **CRL checking**—Since it's not always possible to check a certificate's revocation status against a directory or with the original certificate authority, it is useful for certificates to include information about where to check CRLs.

Eventually, the X.509 v3 specification addressed many of these issues by amending the certificate format to include additional information within a certificate—the version 3 format defines a general format for certificate extensions and specifies a number of standard extensions that can be included the certificate. Thus, the extensions defined for X.509 v3 certificates enable you to associate additional attributes with users or public keys and manage the certification hierarchy. The *Internet X.509 Public Key Infrastructure Certificate and CRL Profile* (see <http://www.ietf.org/rfc/rfc2459.txt>) recommends a set of extensions to be used in Internet certificates (and standard locations for certificate or CA information). These extensions are called *standard extensions*.

The X.509 v3 standard for certificates also suggests that you can define your own extensions and include them in certificates you issue. These extensions are called *private*, *proprietary*, or *custom* extensions and they carry information unique to your organization or business. Keep in mind that applications may not be able to validate certificates that contain private, critical extensions, thus preventing the use of these certificates in a general context.

Before the X.509 v3 standard was finalized, Netscape and other companies had to address some of the most pressing issues listed above with their own extension definitions. For example, Netscape applications (Netscape Navigator 3.0 or higher, and Enterprise Server 2.01 or higher) support an extension known as Netscape Certificate Type Extension that specifies the type of certificate issued, such as client, server, or object signing. Therefore, to maintain compatibility with older versions of browsers that were released before the X.509 v3 specification was finalized, certain kinds of certificates should include some of the Netscape extensions. For details, see “Recommendations for Certificate Extension Use” on page 335.

Note that the X.500 and X.509 specifications are controlled by the International Telecommunication Union (ITU), an international organization that primarily serves large telecom companies, government organizations, and other entities concerned with the international telecommunications network. The Internet Engineering Task Force (IETF), which controls many of the standards that underlie the Internet, is currently developing public-key infrastructure X.509 (PKIX) standards. These proposed standards further refine the X.509 v3 approach to extensions for use on the Internet. The recommendations for certificates and CRLs have reached proposed standard status and are in a document often referred to as PKIX Part 1, which can be retrieved from

<http://www.ietf.org/rfc/rfc2459.txt>.

Some explanations in this appendix also make reference to *Abstract Syntax Notation One (ASN.1)* and *Distinguished Encoding Rules (DER)*. These are specified in the CCITT Recommendations X.208 and X.209. For a quick summary of ASN.1 and DER, see *A Layman’s Guide to a Subset of ASN.1, BER, and DER*, which is available at RSA Laboratories’ web site (<http://www.rsa.com>).

Structure of Certificate Extensions

In RFC 2459, an X.509 certificate extension is defined as follows:

```
Extension ::= SEQUENCE {
    extnID OBJECT IDENTIFIER,
    critical BOOLEAN DEFAULT FALSE,
    extnValue OCTET STRING }
```

Which means, a certificate extension consists of the following:

- The object identifier (OID) for the extension; see Appendix B, “Object Identifiers.”

This identifier uniquely identifies the extension. It also determines the ASN.1 type of value in the value field and how the value is interpreted. That is, when an extension appears in a certificate, the OID appears as the extension ID field (`extnID`) and the corresponding ASN.1 encoded structure appears as the value of the octet string (`extnValue`); see the examples in “Sample Certificate Extensions” on page 335.

- A flag or boolean field called `critical`.

The value, which can be either true or false, assigned to this field indicates whether the extension is critical or noncritical to the certificate.

- If the extension is critical and the certificate is sent to an application that does not understand the extension (based on the extension’s ID), the application must reject the certificate.
 - If the extension is not critical and the certificate is sent to an application that does not understand the extension (based on the extension’s ID), the application can ignore the extension and accept the certificate.
- An octet string containing the DER encoding of the value of the extension.

Typically, the application receiving the certificate checks the extension ID to determine if it can recognize the ID. If it can, it uses the extension ID to determine the type of value used.

Examples of standard extensions defined in the X.509 v3 standard include the following:

- Authority Key Identifier Extension—an extension for identifying the certificate authority’s public key (the key used to sign the certificate).
- Subject Key Identifier Extension—an extension for identifying the subject’s public key (the key being certified).

Note that not all applications support certificates with version 3 extensions. Applications that do support these extensions may not be able to interpret some or all of these specific extensions.

Sample Certificate Extensions

The following is an example of the section of a certificate containing X.509 v3 extensions. (Certificate Management System can display certificates in human-readable format, as shown here.) As shown in the example, certificate extensions appear in sequence and only one instance of a particular extension may appear in a particular certificate; for example, a certificate may contain only one subject key identifier extension. Note that certificates that support these extensions have the version 0x2 (which corresponds to version 3).

```
Certificate:
  Data:
    Version: v3 (0x2)
    ...
  Extensions:
    Identifier: Certificate Type
      Critical: no
      Certified Usage:
        SSL CA
    Identifier: Subject Key Identifier
      Critical: no
      Value:

2c:22:c6:ae:4e:4b:91:c7:fb:4c:cc:ae:84:e8:aa:5b:46:6a:a0:ad
    Identifier: Authority Key Identifier
      Critical: no
      Key Identifier:

2c:22:c6:ae:4e:4b:91:c7:fb:4c:cc:ae:84:e8:aa:5b:46:6a:a0:ad
```

Recommendations for Certificate Extension Use

Most deployments will use some or all of these extensions:

authorityKeyIdentifier. Identifies the public key corresponding to the private key used to sign a certificate.

basicConstraints. Identifies CA certificates and optionally specifies a maximum certificate chain path length.

cRLDistributionPoints. Defines how CRL information for the certificate is to be obtained.

extKeyUsage. Indicates purpose or purposes for which the certificate may be used, either in addition to or instead of the purposes indicated by the keyUsage extension.

keyUsage. Indicates the purpose or purposes for which the public key certified by the certificate may be used.

netscape-cert-type. Indicates the purpose or purposes for which the certificate may be used. Required only for compatibility with some Netscape products that were released before by X.509 v3 was finalized.

subjectAltName. Specifies one or more alternative names for the identity bound by the CA to the certified public key.

subjectKeyIdentifier. Identifies the public key certified by the certificate.

These extensions, plus others, are described in detail in later sections of this appendix. Additional extensions may be useful for a variety of purposes. However, the extensions listed above are either required or recommended for various kinds of certificates issued by Certificate Management System.

Table C-1 summarizes guidelines for using these extensions. The table provides a summary only. Each extension is explained in detail later in the Appendix. Keep the following in mind as you use the table:

- Using certificate extensions incorrectly can lead to severe deployment problems. Make sure you have thoroughly analyzed your deployment needs and completely understand the purpose of each extension you want to use before adding them to certificates.
- Unless otherwise noted in Table C-1, the extensions indicated should be included with certificates of each type to ensure compatibility with both PKIX Part 1 and with future Netscape or iPlanet products.
- Extensions marked “required” must be supported for some existing Netscape or Microsoft products or for other reasons explained in the extension descriptions that follow.

Table C-1 Recommendations for Use of Certificate Extensions with CMS

Certificate type	CA root	Intermediate CA	Issued certificate
SSL client certificate	authorityKeyIdentifier	authorityKeyIdentifier	authorityKeyIdentifier
	basicConstraints: true (required)	basicConstraints: true (required)	
		cRLDistributionPoints	cRLDistributionPoints
	extKeyUsage: client auth	extKeyUsage: client auth	extKeyUsage: client auth
	keyUsage: keyCertSign, cRLSign	keyUsage: keyCertSign, cRLSign	keyUsage: digitalSignature
	netscape-cert-type: SSL CA (if extension exists, bit must be set)	netscape-cert-type: SSL CA (required for client authentication with some Netscape servers)	netscape-cert-type: SSL client (if extension exists, bit must be set; otherwise, not required)
	subjectKeyIdentifier	subjectKeyIdentifier	subjectKeyIdentifier

Table C-1 Recommendations for Use of Certificate Extensions with CMS *(Continued)*

Certificate type	CA root	Intermediate CA	Issued certificate
S/MIME client certificate (single key pair)	authorityKeyIdentifier	authorityKeyIdentifier	authorityKeyIdentifier
		cRLDistributionPoints	cRLDistributionPoints
	extKeyUsage: Email	extKeyUsage: Email	extKeyUsage: Email
	keyUsage: keyCertSign, cRLSign	keyUsage: keyCertSign, cRLSign	keyUsage: digitalSignature
	netscape-cert-type: S/MIME CA (if extension exists, bit must be set)	netscape-cert-type: S/MIME CA (if extension exists, bit must be set)	netscape-cert-type: S/MIME (if extension exists, bit must be set)
			subjectAltName
	subjectKeyIdentifier	subjectKeyIdentifier	subjectKeyIdentifier

Table C-1 Recommendations for Use of Certificate Extensions with CMS *(Continued)*

Certificate type	CA root	Intermediate CA	Issued certificate
S/MIME client certificate (dual key pair)	authorityKeyIdentifier	authorityKeyIdentifier	authorityKeyIdentifier
		cRLDistributionPoints	cRLDistributionPoints
	extKeyUsage: Email	extKeyUsage: Email	extKeyUsage: Email
	keyUsage: keyCertSign, cRLSign	keyUsage: keyCertSign, cRLSign	keyUsage, signing certificate: digitalSignature (required)
			keyUsage, encryption certificate: keyEncipherment (required)
			subjectAltName
	subjectKeyIdentifier	subjectKeyIdentifier	subjectKeyIdentifier

Table C-1 Recommendations for Use of Certificate Extensions with CMS *(Continued)*

Certificate type	CA root	Intermediate CA	Issued certificate
SSL server certificate	authorityKeyIdentifier	authorityKeyIdentifier	authorityKeyIdentifier
		cRLDistributionPoints	cRLDistributionPoints
	extKeyUsage: Server Auth (recommended), Microsoft SGC and Netscape SGC (required for step-up)	extKeyUsage: Server Auth (recommended), Microsoft SGC and Netscape SGC (required for step-up)	extKeyUsage: Server Auth (recommended), Microsoft SGC and Netscape SGC (required for step-up)
	keyUsage: keyCertSign, cRLSign	keyUsage: keyCertSign, cRLSign	keyUsage: keyEncipherment
	netscape-cert-type: SSL CA (if extension exists, bit must be set)	netscape-cert-type: SSL CA (if extension exists, bit must be set)	netscape-cert-type: SSL Client, SSL Server (required for some Netscape servers)
			subjectAltName
	subjectKeyIdentifier	subjectKeyIdentifier	subjectKeyIdentifier

Table C-1 Recommendations for Use of Certificate Extensions with CMS *(Continued)*

Certificate type	CA root	Intermediate CA	Issued certificate
Object signing/ Authenticode certificate	authorityKeyIdentifier	authorityKeyIdentifier	authorityKeyIdentifier
		cRLDistributionPoints	cRLDistributionPoints
	extKeyUsage: Code Signing (required for Authenticode)	extKeyUsage: Code Signing (required for Authenticode)	extKeyUsage: Code Signing (required for Authenticode)
	keyUsage: keyCertSign, cRLSign	keyUsage: keyCertSign, cRLSign	keyUsage: digitalSignature
	netscape-cert-type: Object-signing CA (required for Object Signing)	netscape-cert-type: Object-signing CA (required for Object Signing)	netscape-cert-type: Object-signing (required for Object Signing)
			subjectAltName
	subjectKeyIdentifier	subjectKeyIdentifier	subjectKeyIdentifier

Standard X.509 v3 Certificate Extensions

This section summarizes the extension types that are defined as part of the Internet X.509 Version 3 standard, as of September 1998, and indicates which types are recommended by the PKIX working group.

This section summarizes important information about each certificate. For complete details, see both the X.509 v3 standard (available from the ITU) and the Internet X.509 Public Key Infrastructure - Certificate and CRL Profile (RFC 2459), available at <http://www.ietf.org/rfc/rfc2459.txt>. The descriptions of extensions reference the RFC and section number of the standard draft that discusses the extension; the object identifier (OID) for each extensions is also provided.

Each extension in a certificate can be designated as critical or noncritical. A certificate-using system, such as browser software, must reject the certificate if it encounters a critical extension it does not recognize; however, a noncritical extension can be ignored if it is not recognized.

The descriptions below contain recommendations for use of the extension from Netscape and Microsoft. The Microsoft recommendations were taken from “Structuring X.509 Certificates for Use with Microsoft Products” at <http://www.microsoft.com/security/tech/certificates/structuring.asp>, dated December 4, 1997.

Certificate Management System (CMS) version support is listed for each extension. “Supported” means that the indicated version of CMS ships with built-in support for the extension via a policy plug-in. “Not supported” means that the indicated version of CMS does not ship a policy plug-in for the extension (although the extension can be used if a custom plug-in is written).

These are the standard X.509 v3 extensions described in the sections that follow:

- authorityInfoAccess (page 343)
- authorityKeyIdentifier (page 344)
- basicConstraints (page 345)
- certificatePolicies (page 346)
- cRLDistributionPoints (page 347)
- extKeyUsage (page 348)
- issuerAltName (page 350)
- keyUsage (page 351)
- nameConstraints (page 354)
- OCSPNocheck (page 354)
- policyConstraints (page 355)
- policyMappings (page 356)
- privateKeyUsagePeriod (page 357)
- subjectAltName (page 344)
- subjectDirectoryAttributes (page 359)
- subjectKeyIdentifier (page 360)

authorityInfoAccess

OID

1.3.6.1.5.5.7.1.1

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 4.2.2.1

Criticality

This extension must be noncritical.

Discussion

The Authority Information Access extension indicates how and where to access information about the issuer of the certificate. The extension contains an `accessMethod` and an `accessLocation` field. The `accessMethod` specifies (by an OID) the type and format of information about the issuer found at the `accessLocation`.

PKIX Part 1 defines one `accessMethod` (`id-ad-caIssuers`) to get a list of CAs that have issued certificates higher in the CA chain than the issuer of the certificate using the extension. The `accessLocation` field then typically contains a URL indicating the location and protocol (LDAP, HTTP, FTP) used to retrieve the list.

The Online Certificate Status Protocol (RFC 2560), available at <http://www.ietf.org/rfc/rfc2560.txt>, defines an `accessMethod` (`id-ad-ocsp`) for using OCSP to verify certificates. The `accessLocation` field then contains a URL indicating the location and protocol used to access an OCSP responder that can validate the certificate.

CMS Version Support

Refer to “AuthInfoAccessExt Plug-in Module” on page 136.

- **CMS 4.1:** Not supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

Netscape Recommendation

Netscape recommends that you add this extension with `id-ad-ocsp` and the URL for an OCSP responder to every certificate that can be verified using OCSP.

OCSP signing certificates and CA signing certificates should only use the `authorityInfoAccess` extension to point to an OCSP responder if that responder has been configured to verify them. For example, if there is a hierarchy of responders, a subordinate responder may point to its parent for verification. If a CA signing certificate points to an OCSP responder, that responder's signing certificate should be signed by a different CA (for example, the CA that issued the CA signing certificate in question).

Microsoft Recommendation

Microsoft products do not currently use on-line revocation checking.

authorityKeyIdentifier**OID**

2.5.29.35

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 4.2.1.1

Criticality

This extension is always noncritical and is always evaluated.

Discussion

The Authority Key Identifier extension identifies the public key corresponding to the private key used to sign a certificate. This extension is useful when an issuer has multiple signing keys (for example, due to CA certificate renewal).

The extension consists of either or both of the following:

- an explicit key identifier (`keyIdentifier` field)
- an issuer (`authorityCertIssuer` field) and serial number (`authorityCertSerialNumber` field) identifying a certificate

If the `keyIdentifier` field exists, then it is used to select the certificate with a matching `subjectKeyIdentifier` extension. If the `authorityCertIssuer` and `authorityCertSerialNumber` fields are present, then they are used to identify the correct certificate by issuer and serialNumber.

If this extension is not present, then the issuer name alone is used to identify the issuer certificate.

PKIX Part 1 requires this extension for all certificates except self-signed root CA certificates. Where a key identifier has not been previously established, PKIX recommends that the `authorityCertIssuer` and `authorityCertSerialNumber` fields be specified. These fields permit construction of a complete certificate chain by matching the `SubjectName` and `CertificateSerialNumber` fields in the issuer's certificate against the `authorityCertIssuer` and `authorityCertSerialNumber` in the `AuthorityKeyIdentifier` extension of the subject certificate.

CMS Version Support

Refer to “`AuthorityKeyIdentifierExt` Plug-in Module” on page 144.

- **CMS 4.1:** Supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

Note that Certificate Management System does not use or support the `authorityCertSerialNumber` field in the `Authority Key Identifier` extension.

Netscape Recommendation

Netscape recommends that this extension be present in all certificates and that the `authorityCertIssuer` and `authorityCertSerialNumber` fields be specified. This extension is not supported by Navigator 3.x, but its presence in a certificate won't interfere with Navigator 3.x.

Microsoft Recommendation

Microsoft recommends that this extension be present in all certificates and that the `authorityCertIssuer` and `authorityCertSerialNumber` fields be specified.

basicConstraints

OID

2.5.29.19

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 4.2.1.10

Criticality

PKIX Part 1 requires that this extension be marked critical. This extension is evaluated regardless of its criticality.

Discussion

This extension is used during the certificate chain verification process to identify CA certificates and to apply certificate chain path length constraints. The `ca` component should be set to true for all CA certificates. PKIX recommends that this extension should not appear in end-entity certificates.

If the `pathLenConstraint` component is present, its value must be greater than the number of CA certificates that have been processed so far (starting with the end-entity certificate and moving up the chain). If `pathLenConstraint` is omitted, then all of the higher level CA certificates in the chain must not include this component when the extension is present.

See “CA Certificates and Extension Interactions” on page 371 regarding the interaction of the this extension with the Netscape Certificate Type extension.

CMS Version Support

Refer to “BasicConstraintsExt Plug-in Module” on page 147.

- **CMS 4.1:** Supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

Netscape Recommendation

Netscape requires this extension for all CA certificates.

Microsoft Recommendation

Microsoft recommends this extension for all certificates.

certificatePolicies

OID

2.5.29.32

References

<http://www.ietf.org/rfc/rfc2459.txt> 4.2.1.5

Criticality

This extension may be critical or noncritical.

Discussion

The Certificate Policies extension defines one or more policies, each of which consists of an OID and optional qualifiers. The extension can include a URI to the issuer's Certificate Practice Statement or can embed issuer policy information, such as a user notice in text form. This information can be used by certificate-enabled applications.

If this extension is present, PKIX Part 1 recommends that policies be identified with an OID only, or if necessary only certain recommended qualifiers.

CMS Version Support

Refer to “CertificatePoliciesExt Plug-in Module” on page 151.

- **CMS 4.1:** Not supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

Netscape Recommendation

Netscape recommends that this extension be included at the discretion of the certificate issuer.

Microsoft Recommendation

Microsoft recommends that this extension be included in all certificates.

cRLDistributionPoints**OID**

2.5.29.31

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 4.2.1.14

Criticality

PKIX recommends that this extension be marked noncritical and that it be supported for all certificates.

Discussion

This extension defines how CRL information for this certificate is to be obtained. It should be used if the system is configured to use CRL issuing points.

If the extension contains a `DistributionPointName` of type `URI`, the `URI` is assumed to be a pointer to the current CRL for the associated reasons and will be issued by the associated `cRLIssuer`. The expected values for the `URI` are those defined for the `subjectAltName` extension. If the `distributionPoint` omits reasons, the CRL must include revocations for all reasons. If the `distributionPoint` omits `cRLIssuer`, the CRL must be issued by the CA that issued the certificate.

PKIX recommends that this extension be supported by CAs and applications.

CMS Version Support

Refer to “`CRLDistributionPointsExt` Plug-in Module” on page 166.

- **CMS 4.1:** Supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

Netscape Recommendation

Netscape recommends that this extension be supported for all certificates, with the exception of self-signed root CA certificates.

Microsoft Recommendation

Microsoft recommends that this extension be supported.

extKeyUsage

OID

2.5.29.37

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 4.2.1.13

Criticality

If this extension is marked critical, the certificate must be used for one of the indicated purposes only. If it is not marked critical, it is treated as an advisory field that may be used to identify keys but does not restrict the use of the certificate to the indicated purposes.

Discussion

The Extended Key Usage extension indicates one or more purposes for which the certified public key may be used. These purposes may be in addition to or in place of the basic purposes indicated in the key usage extension.

The Extended Key Usage extension must include OCSP Signing in an OCSP responder's certificate (unless the CA signing key that signed the certificates validated by the responder is also the OCSP signing key). The OCSP responder's certificate must be issued directly by the CA that signs certificates the responder will validate.

The Key Usage, Extended Key Usage, and Basic Constraints extensions act together to define the purposes for which the certificate is intended to be used. Applications can use these extensions to disallow the use of a certificate in inappropriate contexts.

Table C-2 lists the uses defined by PKIX for this extension, and Table C-3 lists uses privately defined by Microsoft or Netscape.

Table C-2 PKIX Extended Key Usage Extension Uses

Use	OID
Server authentication	1.3.6.1.5.5.7.3.1
Client authentication	1.3.6.1.5.5.7.3.2
Code signing	1.3.6.1.5.5.7.3.3
Email	1.3.6.1.5.5.7.3.4
Timestamping	1.3.6.1.5.5.7.3.8
OCSP Signing	1.3.6.1.5.5.7.3.9*

* OCSP Signing is not defined in PKIX Part 1, but in RFC 2560, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP."

Table C-3 Private Extended Key Usage Extension Uses

Use	OID
Certificate trust list signing	1.3.6.1.4.1.311.10.3.1
Microsoft Server Gated Crypto (SGC)	1.3.6.1.4.1.311.10.3.3
Microsoft Encrypted File System	1.3.6.1.4.1.311.10.3.4
Netscape SGC	2.16.840.1.113730.4.1

CMS Version Support

Refer to “ExtendedKeyUsageExt Plug-in Module” on page 171.

- **CMS 4.1:** Not supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

Netscape Recommendations

Netscape recommends that this extension be supported for all certificates, and requires it for all certificates that support step-up, or Server Gated Crypto (SGC). OCSP Signing should be included in all certificates issued to OCSP responders.

Microsoft Recommendations

Microsoft products interpret this extension as follows. If the extension is not present, the certificate is considered to be valid for any usage (to support backward compatibility with certificates that did not use this extension). Otherwise, interpretation depends on usage, as follows:

- Authenticode requires that Code Signing be the unique usage specified.
- SGC operation requires that the SGC usage be specified.
- Timestamping requires that timestamping usage be specified.

Microsoft allows users to control certificate properties that correspond to Extended Key Usage specifications. For example, from the Internet Explorer 4.0 user interface, the user may deselect a CA certificate in a list of CA certificates otherwise trusted for a given usage. Note that the user may only restrict uses, and not add uses that are not supported by the certificate itself. These user settings affect only the interpretation of the certificate on the computer where they are set. They do not affect the certificate itself.

A given certificate is valid only for the intersection of key usages of all the certificates in the chain to its root (as determined by both the Extended Key Usage extension for each certificate and the corresponding user settings). To be valid for a particular usage, the end-entity certificate and all certificates in the chain must all be valid for that usage.

issuerAltName

OID

2.5.29.18

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 4.2.1.8

Criticality

PKIX Part 1 recommends that this extension be marked noncritical.

Discussion

The Issuer Alternative Name extension is used to associate Internet-style identities with the certificate issuer. Names must use the forms defined for subjectAltName.

CMS Version Support

Refer to “IssuerAltNameExt Plug-in Module” on page 184.

- **CMS 4.1:** Not supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

Netscape Recommendation

Netscape products do not examine this extension.

Microsoft Recommendation

Microsoft products do not examine this extension. Microsoft recommends that, for the purposes of building certificate chains, authorityKeyIdentifier be used rather than issuerAltName or the certificate’s issuer name.

keyUsage**OID**

2.5.29.15

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 4.2.1.3

Criticality

This extension may be critical or noncritical. PKIX Part 1 recommends that it should be marked critical if it is used.

Discussion

The Key Usage extension defines the purpose of the key contained in the certificate. The Key Usage, Extended Key Usage, Basic Constraints, and Netscape Certificate Type extensions act together to specify the purposes for which a certificate can be used. For more information on interactions between these extensions in CA certificates, see “CA Certificates and Extension Interactions” on page 371.

If this extension is included at all, set the bits as follows:

- `digitalSignature` (0) for SSL client certificates, S/MIME signing certificates, and object-signing certificates.
- `nonRepudiation` (1) for some S/MIME signing certificates and object-signing certificates. Note, however, that the use of this bit is controversial. You should carefully consider the legal consequences of its use before setting it for any certificate.
- `keyEncipherment` (2) for SSL server certificates and S/MIME encryption certificates.
- `dataEncipherment` (3) when the subjects’s public key is used to encipher user data (as opposed to key material).
- `keyAgreement` (4) whenever the subject’s public key is used for key agreement.
- `keyCertSign` (5) for all CA signing certificates
- `cRLSign` (6) for CA signing certificates that are used to sign CRLs
- `encipherOnly` (7) if the public key is to be used only for enciphering data. If this bit is set, `keyAgreement` should also be set.
- `decipherOnly` (8) if the public key is to be used only for deciphering data. If this bit is set, `keyAgreement` should also be set.

Table C-4 summarizes the above guidelines for typical certificate uses.

Table C-4 Certificate uses and corresponding Key Usage bits

Purpose of certificate	Required Key Usage bit
CA Signing	<code>keyCertSign</code>
	<code>cRLSign</code>
SSL Client	<code>digitalSignature</code>
SSL Server	<code>keyEncipherment</code>
S/MIME Signing	<code>digitalSignature</code>

Table C-4 Certificate uses and corresponding Key Usage bits (*Continued*)

Purpose of certificate	Required Key Usage bit
S/MIME Encryption	keyEncipherment
Certificate Signing	keyCertSign
Object Signing	digitalSignature

If the `keyUsage` extension is present and is marked critical, then it will be used to enforce the usage of the certificate and key. The extension is used to limit the usage of a key; if the extension is not present or not critical, all types of usage are allowed.

If the `keyUsage` extension is present (critical or not), it is used to select from multiple certificates for a given operation. For example, it is used to distinguish separate signing and encryption certificates for users who have separate certificates and key pairs for these operations.

CMS Version Support

Refer to “KeyUsageExt Plug-in Module” on page 189.

- **CMS 4.1:** Supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

Netscape Recommendation

Netscape recommends this extension for all certificates if their intended purpose or purposes are known. Netscape requires this extension for all dual-key signing certificates.

Microsoft Recommendation

Microsoft recommends this extension for all certificates if their intended purpose or purposes are known. If the extension is absent, Microsoft products will assume the certificate is valid for all usages. If the extension is present, Microsoft products will interpret the extension in the same way whether marked critical or not. If the extension is present, the actual usage must conform to the specified usage.

The only Microsoft application that currently enforces this extension is Microsoft Outlook.

nameConstraints

OID

2.5.29.30

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 4.2.1.11

Criticality

PKIX Part 1 requires that this extension be marked critical.

Discussion

This extension, which can be used in CA certificates only, defines a name space within which all subject names in subsequent certificates in a certification path must be located.

CMS Version Support

Refer to “NameConstraintsExt Plug-in Module” on page 202.

- **CMS 4.1:** Not supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

Netscape Recommendation

Netscape products do not currently examine this extension.

Microsoft Recommendation

Microsoft products do not currently examine this extension.

OCSPNocheck

OID

1.3.6.1.5.5.7.48.4

Reference

<http://www.ietf.org/rfc/rfc2560.txt> 4.2.2.2.1

Criticality

This extension should be noncritical.

Discussion

The extension is meant to be included in an OCSP responder's signing certificate. The extension tells an OCSP client that the signing certificate can be trusted without querying the OCSP responder (since the reply would again be signed by the OCSP responder, and the client would again request the validity status of the signing certificate). This extension is null-valued: its meaning is determined by its presence or absence.

Since the presence of this extension in a certificate will cause OCSP clients to trust responses signed with that certificate, use of this extension should be managed carefully. If the OCSP signing key is compromised, the entire process of validating certificates in the PKI will be compromised for the duration of the validity period of the certificate. Therefore, certificates using `OCSPNocheck` should be issued with short lifetimes and be renewed frequently.

CMS Version Support

Refer to "OCSPNoCheckExt Plug-in Module" on page 220.

- **CMS 4.1:** Not supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

Netscape Recommendation

Netscape recommends using this extension in OCSP responder signing certificates. The validity period should be short enough to minimize the potential impact of a compromised OCSP responder signing key to your organization.

Microsoft Recommendation

Microsoft products do not currently use online status checking.

policyConstraints**OID**

2.5.29.36

References

<http://www.ietf.org/rfc/rfc2459.txt> 4.2.1.12

Criticality

This extension may be critical or noncritical.

Discussion

This extension, which is for CA certificates only, constrains path validation in two ways. It can be used to prohibit policy mapping or to require that each certificate in a path contain an acceptable policy identifier.

PKIX requires that, if present, this extension must never consist of a null sequence. At least one of the two available fields must be present.

CMS Version Support

Refer to “PolicyConstraintsExt Plug-in Module” on page 224.

- **CMS 4.1:** Not supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

Netscape Recommendations

Netscape products do not currently examine this extension.

Microsoft Recommendations

Microsoft products do not currently examine this extension.

policyMappings

OID

2.5.29.33

References

<http://www.ietf.org/rfc/rfc2459.txt> 4.2.1.6

Criticality

This extension must be noncritical.

Discussion

The Policy Mappings extension is used in CA certificates only. It lists one or more pairs of OIDs used to indicate that the corresponding policies of one CA are equivalent to policies of another CA. It may be useful in the context of cross-certification.

This extension may be supported by CAs and/or applications.

CMS Version Support

Refer to “PolicyMappingsExt Plug-in Module” on page 227.

- **CMS 4.1:** Not supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

Netscape Recommendation

Netscape products do not currently examine this extension.

Microsoft Recommendations

Microsoft products do not currently examine this extension.

privateKeyUsagePeriod

OID

2.5.29.16

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 4.2.1.4

Discussion

The Private Key Usage Period extension allows the certificate issuer to specify a different validity period for the private key than for the certificate itself. This extension is intended for use with digital signature keys.

PKIX Part 1 recommends against the use of this extension. CAs conforming to PKIX Part 1 *must not* generate certificates with this extension.

CMS Version Support

Refer to “PrivateKeyUsagePeriodExt Plug-in Module” on page 231.

- **CMS 4.1:** Not supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

Netscape Recommendation

Netscape recommends against the use of this extension.

Microsoft Recommendation

Microsoft recommends against the use of this extension.

subjectAltName

OID

2.5.29.17

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 4.2.1.7

Criticality

If the certificate's subject field is empty, this extension must be marked critical.

Discussion

The Subject Alternative Name extension includes one or more alternative (non-X.500) names for the identity bound by the CA to the certified public key. It may be used in addition to the certificate's subject name or as a replacement for it. Defined name forms include Internet electronic mail address (SMTP, as defined in RFC-822), DNS name, IP address, and uniform resource identifier (URI).

PKIX requires this extension for entities that are identified by name forms other than the X.500 distinguished name (DN) used in the subject field. PKIX Part 1 describes additional rules for the relationship between this extension and the subject field.

Email addresses may be provided either in the Subject Alternative Name extension, the certificate subject name field, or both. If the email address is provided as part of the subject name, it must be in the form of the `EmailAddress` attribute defined by PKCS-9. Software that supports S/MIME must be able to read an email address from either the Subject Alternative Name extension or from the subject name field.

CMS Version Support

Refer to "SubjectAltNameExt Plug-in Module" on page 235.

- **CMS 4.1:** Supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

Netscape Recommendation

Netscape recommends the use of this extension with all certificates issued by a CA (except for SSL client certificates).

Netscape products read only the first alternative name in this extension, and ignore the rest. For S/MIME certificates, Netscape software first checks the first alternative name in this extension (if the extension is present) for the `EmailAddress` attribute. If the first alternative name is not an `EmailAddress` attribute, Netscape software looks for the `e=` attribute of the DN. If the `e=` attribute is not present, Netscape software looks for the `mail=` attribute of the DN.

Microsoft Recommendation

Microsoft recommends the use of this extension whenever X.500 guidelines are insufficient for naming purposes. Currently, no Microsoft products require the use of Subject Alternative Name. All Microsoft products that support S/MIME are capable of reading email names from this extension or from the subject name. Future versions of Microsoft Exchange Server will issue certificates with X.500 names that do not contain the Email Address attribute, and will place the SMTP address in the Subject Alternative Name extension.

subjectDirectoryAttributes

OID

2.5.29.9

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 4.2.1.9

Criticality

PKIX Part 1 requires that this extension be marked noncritical.

Discussion

The Subject Directory Attributes extension conveys any desired directory attribute values for the subject of the certificate. It is not recommended as an essential part of the proposed PKIX standard, but may be used in local environments.

CMS Version Support

Refer to “SubjectDirectoryAttributesExt Plug-in Module” on page 241.

- **CMS 4.1:** Not supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

Netscape Recommendation

Netscape products do not examine this extension.

Microsoft Recommendation

Microsoft products do not examine this extension.

subjectKeyIdentifier

OID

2.5.29.14

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 4.2.1.2

Criticality

This extension is always noncritical.

Discussion

The Subject Key Identifier extension identifies the public key certified by this certificate. This extension provides a way of distinguishing public keys if more than one is available for a given subject name, for example after the certificate has been renewed with a new key.

The value of this extension should be calculated by performing a SHA-1 hash of the certificate's DER-encoded `subjectPublicKey`, as recommended by PKIX. The Subject Key Identifier extension is used in conjunction with the Authority Key Identifier extension for CA certificates. If the CA certificate has a Subject Key Identifier extension, the key identifier in the Authority Key Identifier extension (of the certificate being verified) should match the key identifier of the CA's Subject Key Identifier extension. It is not necessary for the verifier to recompute the key identifier in this case.

PKIX Part 1 requires this extension for all CA certificates and recommends it for all other certificates.

CMS Version Support

Refer to "SubjectKeyIdentifierExt Plug-in Module" on page 245.

- **CMS 4.1:** Supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

Netscape Recommendation

Netscape recommends this extension for all certificates.

Microsoft Recommendation

Microsoft recommends this extension for all certificates.

Introduction to CRL Extensions

Since its initial publication, the X.509 standard for CRL formats has been amended to include additional information within a CRL. Version 2, the latest version, allows you to add information as CRL extensions.

The extensions defined by ANSI X9 and ISO/IEC/ITU for X.509 v2 CRLs [X.509] [X9.55] enable you to associate additional attributes with CRLs. The *Internet X.509 Public Key Infrastructure Certificate and CRL Profile* (see <http://www.ietf.org/rfc/rfc2459.txt>) recommends a set of extensions to be used in CRLs. These extensions are called *standard CRL extensions*.

The standard also suggests that you can define your own extensions and include them in CRLs you issue. These extensions are called *private*, *proprietary*, or *custom* CRL extensions and they carry information unique to your organization or business. Keep in mind that applications may not be able to validate CRLs that contain private, critical extensions, thus preventing the use of these CRLs in a general context.

NOTE Some explanations in this chapter make reference to Abstract Syntax Notation One (ASN.1) and Distinguished Encoding Rules (DER). These are specified in the CCITT Recommendations X.208 and X.209. For a quick summary of ASN.1 and DER, see *A Layman's Guide to a Subset of ASN.1, BER, and DER*, which is available at RSA Laboratories' web site (<http://www.rsa.com>).

Structure of CRL Extensions

A CRL extension consists of the following:

- The object identifier (OID) for the extension; see Appendix B, “Object Identifiers.”

This identifier uniquely identifies the extension. It also determines the ASN.1 type of value in the value field and how the value is interpreted. That is, when an extension appears in a CRL, the OID appears as the extension ID field (`extnID`) and the corresponding ASN.1 encoded structure appears as the value of the octet string (`extnValue`); see the examples in “Sample Certificate Extensions” on page 335.

- A flag or boolean field called `critical`.

The `true` or `false` value assigned to this field indicates whether the extension is critical (true) or noncritical (false) to the CRL.

- If the extension is critical and the CRL is sent to an application that does not understand the extension (based on the extension's ID), the application must reject the CRL.
 - If the extension is not critical and the CRL is sent to an application that does not understand the extension (based on the extension's ID), the application can ignore the extension and accept the CRL.
- An octet string containing the DER encoding of the value of the extension.

Typically, the application receiving the CRL checks the extension ID to determine if it can recognize the ID. If it can, it uses the extension ID to determine the type of value used.

Sample CRL and CRL Entry Extensions

The following is an example of the section of a CRL containing X.509 v2 extensions. (Certificate Management System can display CRLs in human-readable format, as shown here.) As shown in the example, CRL extensions appear in sequence and only one instance of a particular extension may appear in a particular CRL; for example, a CRL may contain only one authority key identifier extension. However, CRL-entry extensions appear in appropriate entries in the CRL.

```
Certificate Revocation List:
  Data:
    Version:  v2
    ...
  Extensions:
    Identifier: Authority Key Identifier
    Critical: no
    Key Identifier:

2c:22:c6:ae:4e:4b:91:c7:fb:4c:cc:ae:84:e8:aa:5b:46:6a:a0:ad
  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
    ...
```

Standard X.509 v3 CRL Extensions

In addition to certificate extensions, the X.509 v3 proposed standard defines extensions to CRLs, which provide methods for associating additional attributes with Internet CRLs. These are of two kinds: extensions to the CRL itself, and extensions to individual certificate entries in the CRL.

- Extensions for CRLs
- CRL Entry Extensions

Extensions for CRLs

The sections that follow describe the CRL extension types that are defined as part of the Internet X.509 v3 Public Key Infrastructure proposed standard, as of September 1998.

These are the CRL extensions described in the sections that follow:

- `authorityKeyIdentifier` (page 364)
- `CRLNumber` (page 365)
- `deltaCRLIndicator` (page 365)
- `issuerAltName` (page 366)
- `issuingDistributionPoint` (page 366)

`authorityKeyIdentifier`

OID

2.5.29.35

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 5.2.1

Discussion

The Authority Key Identifier extension for a CRL identifies the public key corresponding to the private key used to sign the CRL. For details, see the discussion under certificate extensions at `authorityKeyIdentifier`.

CMS Version Support

Refer to “`AuthorityKeyIdentifier` Rule” on page 285.

- **CMS 4.1:** Not supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

CRLNumber

OID

2.5.29.20

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 5.2.3

Criticality

This extension must not be critical.

Discussion

The CRL Number extension specifies a sequential number for each CRL issued by a CA. It allows users to easily determine when a particular CRL supersedes another CRL.

PKIX requires that all CRLs have this extension.

CMS Version Support

Refer to “CRLNumber Rule” on page 287.

- **CMS 4.1:** Not supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

deltaCRLIndicator

OID

2.5.29.27

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 5.2.4

Criticality

PKIX requires that this extension be critical if it exists.

Discussion

The Delta CRL Indicator extension identifies a delta-CRL. The use of delta-CRLs allows changes to be added to the local database while ignoring unchanged information that is already in the local database. This can significantly improve processing time for applications that store revocation information in a format other than the CRL structure.

This extension is used only with delta-CRLs, which are not supported by Certificate Management System.

CMS Version Support

- **CMS 4.1:** Not supported
- **CMS 4.2:** Not supported
- **CMS 4.2-SP2:** Supported

issuerAltName

OID

2.5.29.18

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 5.2.2

Discussion

The Issuer Alternative Name extension allows additional identities to be associated with the issuer of the CRL. For details, see the discussion under certificate extensions at `issuerAltName`.

CMS Version Support

Refer to “IssuerAlternativeName Rule” on page 293.

- **CMS 4.1:** Not supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

issuingDistributionPoint

OID

2.5.29.28

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 5.2.5

Criticality

PKIX requires that this extension be critical if it exists.

Discussion

The Issuing Distribution Point CRL extension identifies the CRL distribution point for a particular CRL and indicates what kinds of revocation it covers.

PKIX Part I does not require this extension.

CMS Version Support

Refer to “IssuingDistributionPoint Rule” on page 297.

- **CMS 4.1:** Not supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

CRL Entry Extensions

The sections that follow lists the CRL entry extension types that are defined as part of the Internet X.509 v3 Public Key Infrastructure proposed standard, as of September 1998. All of these extensions are noncritical.

These are the CRL entry extensions described in the sections that follow:

- `certificateIssuer` (page 367)
- `holdInstructionCode` (page 368)
- `invalidityDate` (page 368)
- `reasonCode` (page 369)

`certificateIssuer`

OID

2.5.29.29

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 5.3.4

Discussion

The Certificate Issuer extension identifies the certificate issuer associated with an entry in an indirect CRL.

This extension is used only with indirect CRLs, which are not supported by Certificate Management System.

CMS Version Support

- **CMS 4.1:** Not supported
- **CMS 4.2:** Not supported
- **CMS 4.2-SP2:** Not supported

holdInstructionCode

OID

2.5.29.23

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 5.3.2

Discussion

The Hold Instruction Code extension indicates the action to be taken after encountering a certificate that has been placed on hold.

CMS Version Support

Refer to “HoldInstruction Rule” on page 290.

- **CMS 4.1:** Not supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

invalidityDate

OID

2.5.29.24

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 5.3.3

Discussion

The Invalidity Date extension provides the date on which the private key was compromised or that the certificate otherwise became invalid.

CMS Version Support

Refer to “InvalidityDate Rule” on page 291.

- **CMS 4.1:** Not supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

reasonCode

OID

2.5.29.21

Reference

<http://www.ietf.org/rfc/rfc2459.txt> 5.3.1

Discussion

The Reason Code extension identifies the reason for certificate revocation.

CMS Version Support

Refer to “CRLReason Rule” on page 288.

- **CMS 4.1:** Not supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

Netscape-Defined Certificate Extensions

Netscape has defined certain certificate extensions for use with Navigator and Communicator. Some of the extensions that have been defined are now obsolete, and others can be superseded by the extensions defined in the X.509 proposed standard. All Netscape extensions should be tagged as noncritical, so that their presence in a certificate does not make that certificate incompatible with other clients.

The specifications for all Netscape-defined extensions are defined at <http://home.netscape.com/eng/security/comm4-cert-exts.html>. For most CMS deployments, only `netscape-cert-type` and `netscape-comment` need to be supported to maintain compatibility with Navigator 3.x. Therefore, only these two Netscape certificate extensions are described here.

`netscape-cert-type`

OID

2.16.840.1.113730.1

Discussion

The Netscape Certificate Type extension can be used to limit the purposes for which a certificate can be used. It has been replaced by the X.509 v3 extensions `extKeyUsage` and `basicConstraints`, but must still be supported in deployments that include Navigator 3.x clients.

If the extension exists in a certificate, it limits the certificate to the uses specified in it. If the extension is not present, the certificate can be used for all applications except object signing.

The value is a bit-string, where the individual bit positions, when set, certify the certificate for particular uses as follows:

- bit 0: SSL Client certificate
- bit 1: SSL Server certificate
- bit 2: S/MIME certificate
- bit 3: Object-signing certificate
- bit 4: Reserved for future use
- bit 5: SSL CA certificate
- bit 6: S/MIME CA certificate
- bit 7: Object-signing CA certificate

CMS Version Support

Refer to “NSCertTypeExt Plug-in Module” on page 215.

- **CMS 4.1:** Supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

netscape-comment

OID

2.16.840.1.113730.13

Discussion

The value of this extension is an IA5String. It is a comment that can be displayed to the user when the certificate is viewed.

CMS Version Support

Refer to “NSCCommentExt Plug-in Module” on page 211.

- **CMS 4.1:** Not supported
- **CMS 4.2:** Supported
- **CMS 4.2-SP2:** Supported

CA Certificates and Extension Interactions

Netscape recommends that all CA certificates contain the `basicConstraints` extension, as this is the standard way to identify a CA certificate. In addition, to ensure support for Navigator 3.x, CAs should also use `netscape-cert-type`. These two extensions can interact with each other. The following table describes what different combinations of the two extensions mean.

Extensions Present	Description
Only <code>basicConstraints</code>	The certificate is a CA certificate if the <code>cA</code> component is true. Path length processing is done as described above.
Only <code>netscape-cert-type</code>	The certificate is a CA if at least one of the CA bits is set: SSL CA (5), S/MIME CA (6), or object-signing CA (7). The certificates issued by this CA are limited to the particular applications specified. Path length processing is done as though the <code>pathLenConstraint</code> is unlimited.
Neither extension	The certificate is not a CA.

Extensions Present	Description
Both extensions	The certificate is a CA certificate if the <code>cA</code> component of <code>basicConstraints</code> is true. If one or more of the SSL CA (5), S/MIME CA (6), or object-signing CA (7) bits are set in the <code>netscape-cert-type</code> extension, then the CA will be limited to issuing certificates for the specified application areas; otherwise, the CA can issue certificates for any application.

A certificate chain generally consists of an entity certificate, zero or more intermediate CA certificates, and a root CA certificate. Typically the root CA certificate is self-signed and is loaded into Communicator's certificate database as a trusted CA.

An exchange of certificates takes place when performing an SSL handshake, when sending an S/MIME message, or when sending a signed object. As part of the handshake, the sender is expected to send the subject certificate and any intermediate CA certificates needed to link the subject certificate to the trusted root. For certificate chaining to work properly the certificates should have the following properties:

- CA certificates must have either the `basicConstraints` extension, the `netscape-cert-type` extension with one or more CA bits set, or both, as described above.
- If CAs issue multiple certificates for the same identity, for example for separate signing and encryption keys, they must include the `keyUsage` extension in the subject certificates.
- If CAs ever intend to generate new keys for their CA, they must add the `authorityKeyIdentifier` extension to all subject certificates. If the `key ID` is anything other than the SHA-1 hash of the CA certificates `subjectPublicKeyInfo` field, then the CA certificate should contain the `subjectKeyIdentifier` extension. This will allow for a smooth transition when the new issuing certificate becomes active.

Index

A

- adding extensions
 - to CRLs 284
 - to end-entity certificates 132
- adding new directory attributes 319
- Attribute Present Constraints policy 90
- Audit log
 - configuring 303
 - logging to Windows NT event log 308
- authentication
 - automated vs. manual 20
 - built-in modules 20
 - list of 21
 - NISAuth 39
 - PortalEnroll 44, 47
 - See also PIN Generator tool
 - UidPwdDirAuth 26
 - UidPwdPinDirAuth 31
 - configuring for end-user enrollment 22
 - default forms for users 22
 - directory- and PIN-based 30
 - directory-based 24
 - during certificate renewal 22
 - during certificate revocation 22
 - how to write custom plug-ins 22
 - manual 23
 - NIS server-based 37
- Authority Information Access extension policy 136
- Authority Key Identifier extension policy 144
- authorityKeyIdentifier 344, 364, 372
- automated enrollment 20

B

- base DN 315
- Basic Constraints extension policy 147
- basicConstraints 345, 371
- built-in plug-in modules
 - See plug-in modules
- bulk enrollment 56

C

- CA certificate mapper 255
- CA certificate publisher 275
- Certificate Manager
 - enrollment forms for 60
 - logging to Windows NT event log 308
- Certificate Policy extension policy 151
- certificate renewal
 - validity period for 106
- Certificate Renewal Window extension policy 156
- Certificate Scope of Use extension policy 161
- certificate-based enrollment 53
 - forms for 54
 - what you need 54
 - when to use 53
- certificateIssuer 367
- certificatePolicies 346
- certificates
 - enrollment forms 57
 - automated 57

- manual 57
 - extensions for 331–372
- challenge password 22
- changing
 - DER encoding order of DirectoryString 322
- Chapter Single Template 313, 329
- client certificates
 - for DSA key pairs 61
- CMC request enrollment 61
- common features in extension policies 135
- constraints-specific policies
 - attribute present constraints 90
 - DSA key constraints 95
 - issuer constraints 98
 - key algorithm constraints 101
 - renewal constraints 103
 - renewal validity constraints 106
 - revocation constraints 110
 - RSA key constraints 112
 - signing algorithm constraints 115
 - subordinate CA name constraints 118
 - unique subject name constraints 121
 - validity constraints 124
- constraints-specific policy modules 88
- conventions used in this book 14
- CRL Distribution Point extension policy 166
- CRL extension modules
 - AuthorityKeyIdentifier 285
 - CRLNumber 287
 - CRLReason 288
 - HoldInstruction 290
 - InvalidityDate 291
 - IssuerAlternativeName 293
 - IssuingDistributionPoint 297
 - list of 285
- CRL publisher 279
- cRLDistributionPoints 347
- CRLNumber 365
- CRLs
 - extensions for 364–369
 - extension-specific modules 361
 - supported versions 283
- custom plug-ins
 - for authentication 22
 - for logs 302

- for mapping directory entries 254
- for policy 135
- for publishing to a directory 273

D

- Data Recovery Manager
 - logging to Windows NT event log 308
- defining custom OIDs 329
- deltaCRLIndicator 365
- DER-encoding order of DirectoryString 322
- directory
 - removing expired certificates from 76
- directory attributes
 - adding new 319
 - supported in CMS 316
- directory-based authentication 24
 - user ID and password 24
 - user ID, password, and PIN 30
- distinguished name (DN)
 - base DN 315
 - characters allowed in CMS 316
 - components 314
 - defined 313
 - extending attribute support 318
 - guidelines for choosing DNs 325
 - role in certificates 323
 - CA certificates 324
 - end-entity certificates 324
 - root DN 315
- DN character support in CMS 316
- DN components mapper 259, 264
- DN pattern mapper 265
- documentation
 - conventions followed 14
 - where to find 15
- DSA client certificates 61
- DSA Key Constraints policy 95
- DSA key pairs 61

E

- encrypted file system (EFS) 172
- end-entity certificate publisher 277
- end-entity enrollment forms 57
 - automated 57
 - manual 57
- end-entity forms
 - for enrollment 59
- enrollment
 - automated 20
 - in bulk 56
 - manual 20
- enrollment forms
 - for Certificate Managers 60
 - for end users 59
 - for object signing certificates 61
 - for OCSP responder certificates 60
 - for Registration Managers 60
 - for servers 60
 - generating DSA key pairs 61
- Error log
 - configuring 303
- event log
 - configuring 308
 - logging audit and system messages 308
- expired certificates
 - removing from the directory 76
- Extended Key Usage extension policy 171
 - OIDs for encrypted file system 172
- extending directory-attribute support in CMS 318
- extensions 331–372
 - 134
 - adding to end-entity certificates 132
 - an example 335
 - authorityKeyIdentifier 344, 364, 372
 - basicConstraints 345, 371
 - CA certificates and 371–372
 - certificateIssuer 367
 - certificatePolicies 346
 - cRLDistributionPoints 347
 - CRLNumber 365
 - deltaCRLIndicator 365
 - extKeyUsage 348
 - holdInstructionCode 368
 - introduction to 332
 - invalidityDate 368
 - issuerAltName 350, 366
 - issuingDistributionPoint 366
 - keyUsage 351
 - nameConstraints 354
 - netscape-cert-type 370, 371
 - netscape-comment 371
 - Netscape-defined 369–372
 - policyConstraints 355
 - policyMappings 356
 - privateKeyUsagePeriod 357
 - reasonCode 369
 - recommendations for usage 335–341
 - structure of 334
 - subjectAltName 358
 - subjectDirectoryAttributes 359
 - subjectKeyIdentifier 360
 - X.509 certificate, summarized 341–361
 - X.509 CRL, summarized 364–369
- extension-specific policies
 - authority information access 136
 - authority key identifier 144
 - basic constraints 147
 - certificate policy 151
 - certificate renewal window 156
 - certificate scope of use 161
 - common features 135
 - CRL distribution point 166
 - extended key usage 171
 - Generic ASN.1 177
 - issuer alternative name 184
 - key usage 189
 - name constraints 202
 - Netscape certificate comment 211
 - Netscape certificate type 215
 - policy constraints 220, 224
 - policy mappings 227
 - private key usage period 231
 - remove basic constraints 233
 - subject alternative name 235
 - subject directory attributes 241
 - subject key identifier 245
- extension-specific policy modules 332
 - list of 134
- extKeyUsage 348

F

- file-based logging
 - configurable parameters 304
 - plug-in module name 304
- file-based publisher 274
- fonts used in this book 14

G

- Generic ASN.1 extension policy 177

H

- holdInstructionCode 368
- HTML forms
 - for end entities
 - for enrollment 59

I

- invalidityDate 368
- Issuer Alternative Name extension policy 184
- Issuer Constraints policy 98
- issuerAltName 350, 366
- issuingDistributionPoint 366

J

- jobs
 - built-in modules 67
 - RenewalNotificationJob 68, 69
 - RequestInQueueJob 68, 73
 - UnpublishExpiredJob 68, 76
 - compared to plug-in implementation 69
 - specifying schedule for 80

K

- Key Algorithm Constraints policy 101
- Key Usage extension policy 189
- keyUsage 351

L

- listing
 - of CRL extension modules 285
 - of schedulable jobs 68
- locating directory entries for publishing
 - how to write custom plug-ins 254
- location of
 - CMS documentation 15
- logging
 - built-in modules
 - file 303, 304
 - list of 302
 - NTEventLog 309
 - how to write custom plug-ins 302
 - to files 303

M

- manual authentication 23
- manual enrollment 20
- mapper modules
 - introduction 251, 271
 - list of 253
- mappers
 - created during installation 255, 266
 - defined 251, 271
- mappers that use
 - CA certificate 255
 - DN components 259
 - DN patterns 265
 - subject attributes 268
 - subject names 264
- mapping certificates to directory entries 252
- message templates for notifications 81

N

- Name Constraints extension policy 202
- nameConstraints 354
- Netscape Certificate Comment extension policy 211
- Netscape Certificate Type extension policy 215
- netscape-cert-type 370, 371
- netscape-comment 371
- NIS server-based authentication 37
 - configurable parameters 39
 - plug-in module name 39
- notifications
 - customizing 81
 - templates 83
 - sending renewal notifications to end entities 69
 - to agents about pending requests 73
 - to agents about unpublishing certificates 76
- NT Event log
 - plug-in module name 309

O

- object identifiers 329
- object signing certificates
 - for third-party tools 63
 - how to enroll for 61
- OCSP publisher 281
- OCSP responder certificates
 - how to enroll for 60
- OIDs 329
- overview
 - authentication modules 20

P

- plug-in modules
 - for authentication
 - list of 21
 - NISAuth 39
 - PortalEnroll 47
 - UidPwdDirAuth 26

- UidPwdPinDirAuth 31
- for CRL extensions
 - AuthorityKeyIdentifier 285
 - CRLNumber 287
 - CRLReason 288
 - HoldInstruction 290
 - InvalidityDate 291
 - IssuerAlternativeName 293
 - IssuingDistributionPoint 297
 - list of 285
- for logging to file 304
- for logging to NT Event log 309
- for logs
 - list of 302
- for policy 87, 131, 329
 - AttributePresentConstraints 90
 - AuthInfoAccessExt 136
 - AuthorityKeyIdentifierExt 144
 - BasicConstraintsExt 147
 - CertificatePoliciesExt 151
 - CertificateRenewalWindowExt 156
 - CertificateScopeOfUseExt 161
 - CRLDistributionPointsExt 166
 - DSAKeyConstraints 95
 - ExtendedKeyUsageExt 171
 - GenericASN1Ext 177
 - IssuerAltNameExt 184
 - IssuerConstraints 98
 - KeyAlgorithmConstraints 101
 - KeyUsageExt 189
 - NameConstraintsExt 202
 - NSCCommentExt 211
 - NSCertTypeExt 215
 - OCSPNoCheckExt 220
 - PolicyConstraintsExt 224
 - PolicyMappingsExt 227
 - PrivateKeyUsagePeriodExt 231
 - RemoveBasicConstraintsExt 233
 - RenewalConstraints 103
 - RenewalValidityConstraints 106
 - RevocationConstraints 110
 - RSAKeyConstraints 112
 - SigningAlgorithmConstraints 115
 - SubCANameConstraints 118
 - SubjectAltNameExt 235
 - SubjectDirectoryAttributesExt 241
 - SubjectKeyIdentifierExt 245

- UniqueSubjectNameConstraints 121
- ValidityConstraints 124
- for publishing 272
 - FileBasedPublisher 274
 - LdapCaCertPublisher 275
 - LdapCaSimpleMap 255
 - LdapCriPublisher 279
 - LdapDNCompsMap 259
 - LdapDNExactMap 264
 - LdapSimpleMap 265
 - LdapSubjAttrMap 268
 - LdapUserCertPublisher 277
 - list of 253, 273
 - OCSPPublisher 281
- for scheduling jobs
 - list of 68
 - RenewalNotificationJob 69
 - RequestInQJob 73
 - UnpublishExpiredJob 76
- policy
 - built-in plug-in modules 87, 131, 329
 - constraints-specific modules 88
 - extension-specific modules 332
 - how to write custom plug-ins 135
- Policy Constraints extension policy 220, 224
- Policy Mappings extension policy 227
- policyConstraints 355
- policyMappings 356
- portal enrollment 44
 - configurable parameters 47
 - plug-in module name 47
- PQG parameters 61
- Private Key Usage Period extension policy 231
- privateKeyUsagePeriod 357
- publisher modules
 - introduction 272
 - list of 273
- publishers
 - created during installation 275, 277, 279
- publishers that can publish to
 - CA's entry in the directory 275, 279
 - files 274
 - OCSP responder 281
 - users' entries in the directory 277
- publishing

- how to write custom plug-ins 273
- publishing certificates and CRLs to directory entries 272

R

- reasonCode 369
- registering
 - custom OIDs 330
- Registration Manager
 - enrollment forms for 60
 - logging to Windows NT event log 308
- Remove Basic Constraints extension policy 233
- Renewal Constraints policy 103
- Renewal Validity Constraints policy 106
- Revocation Constraints policy 110
- root DN 315
- RSA Key Constraints policy 112

S

- server enrollment forms 60
- setting CRL extensions 284
- Signing Algorithm Constraints policy 115
- Subject Alternative Name extension policy 235
- subject attribute mapper 268
- Subject Directory Attributes extension policy 241
- Subject Key Identifier extension policy 245
- subjectAltName 358
- subjectDirectoryAttributes 359
- subjectKeyIdentifier 360
- subordinate CA
 - enrollment forms for 60
- Subordinate CA Name Constraints policy 118
- Sun ONE 11
- support for DN characters in CMS 316
- System log
 - configuring 303
 - logging to Windows NT event log 308

T

templates

- for notifications 81

- customizing 83

- token list 83

- templates

- for automated notifications 81

type styles used in this book 14

U

Unique Subject Name Constraints policy 121

user enrollment forms 59

user ID and password based authentication 24

- configurable parameters 26

- plug-in module name 26

user ID, password, and PIN based authentication 30

- configurable parameters 31

- module name 31

V

Validity Constraints policy 124

W

Windows NT event log

- logging audit and system messages 308

wireless certificates 60

