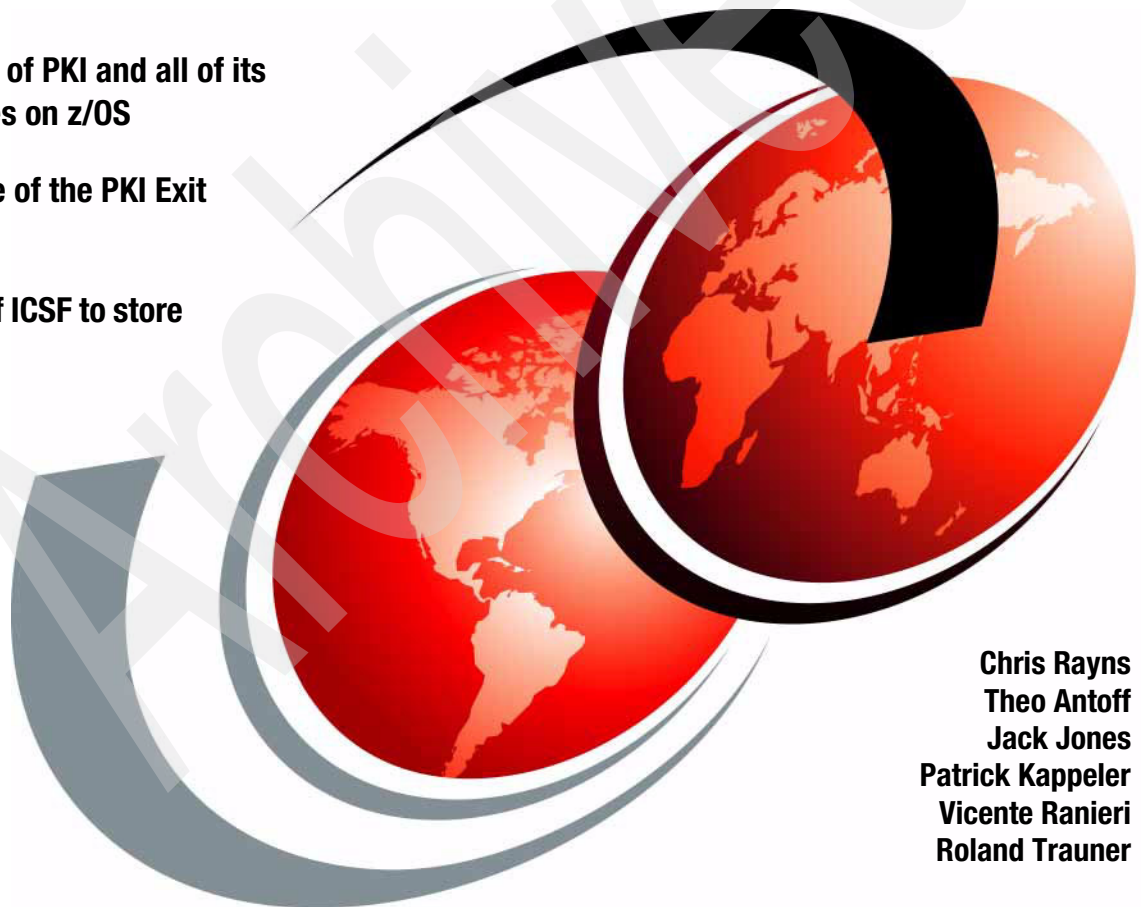


Implementing PKI Services on z/OS

Installation of PKI and all of its prerequisites on z/OS

An example of the PKI Exit

PKI's use of ICSF to store Master Key



Chris Rayns
Theo Antoff
Jack Jones
Patrick Kappeler
Vicente Ranieri
Roland Trauner



International Technical Support Organization

Implementing PKI Services on z/OS

February 2004

Archived

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (February 2004)

This edition applies to z/OS Version 1, Release 3.

© Copyright International Business Machines Corporation 2004. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	vii
Trademarks	viii
 Preface	ix
The team that wrote this redbook	ix
Become a published author	x
Comments welcome	xi
 Chapter 1. Security Server PKI Services	1
1.1 Overview of digital certificate	2
1.2 The PKIX standards	4
1.2.1 CA hierarchy	6
1.2.2 The X.509 certificate and Certificate Revocation List	9
1.2.3 The x.509 v3 certificate extension fields	14
1.2.4 Certificate and CRL appearance	17
1.3 The z/OS PKI Services	21
1.3.1 Security Server PKI Services in z/OS	21
1.3.2 Prerequisite products	22
1.3.3 Requests supported by z/OS PKI Services	23
1.3.4 Browser and server certificates	24
1.3.5 The z/OS PKI Services architecture	26
1.4 Security Server PKI Services enhancement in z/OS V1R4	29
1.4.1 Sysplex support	30
1.4.2 Event notification via e-mail	32
1.4.3 Additional distinguished name qualifier support	33
1.4.4 LDAP password encryption	33
1.4.5 PKCS#7 certificate chain support	33
1.4.6 Key generation via PCICC	35
1.4.7 Additional default CERTAUTH	35
1.4.8 Summary of z/OS PKI external characteristics as of z/OS V1R4	35
 Chapter 2. RACF for PKI Services	37
2.1 Introduction to creating an RACF environment for new products	38
2.1.1 RACF group structure	38
2.1.2 Machine user IDs	39
2.1.3 System data set profiles	40
2.1.4 Ownership	40
2.2 New RACF features	40
2.2.1 Access control lists	41

2.2.2 Automatic assignment of UID/GID	50
2.3 Setting up RACF environment for PKI prerequisites	55
2.3.1 z/OS UNIX level security	56
2.3.2 RACF for Web server	58
2.3.3 RACF for OCSF and OCEP	59
2.3.4 RACF for LDAP	59
2.3.5 RACF for ICSF	60
2.4 Setting up the RACF environment for PKI Services	61
2.4.1 Add RACF groups for PKI Services	61
2.4.2 Adding RACF user IDs for PKI Services	62
2.4.3 Adding PKI data set profiles	63
2.4.4 Using RACF to create certificates	64
2.4.5 Daemon and server control for PKI user ID and surrogate user ID ..	76
2.4.6 Allow PKI user ID to act as CA	76
2.4.7 Allow Web server to access its own key ring	77
2.4.8 Allow Web server user ID to switch identity to surrogate user ID ...	77
2.4.9 Profile for PKI Services procedure in class STARTED	77
2.4.10 Allow access for PKISTU to OCSF	77
2.4.11 ICSF	77
2.4.12 Protect certificate functions	79
2.5 RACF administration for PKI Services	84
2.5.1 Creating a help desk function	84
2.5.2 Administering certificates with the HostIdMappings extension	85
2.5.3 Display your PKI Services certificates	86
2.5.4 Establishing PKI Services as intermediate certificate authority	89
2.5.5 Renewing your PKI Services CA certificate	91
2.5.6 Recovering a CA certificate profile	91
2.5.7 Controlling applications that call R_PKIServ	94
2.5.8 Using encrypted passwords for LDAP servers	97
2.5.9 Register a Personal Certificate with RACF	100
Chapter 3. Easy steps to get PKI up and running	105
3.1 Preparing the PKI Server installation	106
3.1.1 Steps to set up the PKI server	106
3.2 Prepare and configure the environment	106
3.3 Setting up the Web servers for PKI	107
3.3.1 Why do we need two Web servers?	108
3.3.2 Setting up the Web server as a secure Web server	108
3.3.3 Customizing the Web server for SSL	108
3.3.4 Customizing the first Web server for PKI	109
3.3.5 Customizing the second Web server for PKI	113
3.4 Setting up the LDAP server for PKI	116
3.4.1 LDAP setup: running the ldapcnf utility	119

3.5	Setting up the PKI Services task	126
3.6	Configure OCSF and OCEP to work with PKI Services	127
3.7	Configure the PKI Services	128
3.7.1	Set up the environment variables for PKI Services	129
3.7.2	Customizing the PKI Services configuration file	130
3.7.3	Customizing the PKI template	132
3.8	Checking the VSAM data set	134
Chapter 4. Customizing the z/OS PKI Services: the template file		137
4.1	The template file, CGI, and the Web end user	138
4.1.1	The template file sections	138
4.1.2	The CGI modules	151
4.1.3	Relationship between CGI modules and Web user templates	153
4.1.4	An example of simple customization of the template file	155
4.2	Structure of the template file for interaction with the PKI Administrator	158
4.2.1	The CGI modules	158
4.2.2	Customization of the administration Web pages	160
4.2.3	PKI administrator e-mail address	161
4.2.4	PKI Services certification policy	161
4.2.5	Link to PKI Services from your home page	162
4.2.6	Certificate authentication for administrators	163
Chapter 5. PKI Installation using the IKYSETUP REXX exec		175
5.1	IKYSETUP overview	176
5.2	IKYSETUP variables	176
5.2.1	Compulsory changes to IKYSETUP	177
5.2.2	Probable changes to IKYSETUP	179
5.2.3	Optional changes to IKYSETUP	184
Chapter 6. PKI Exit		205
6.1	PKI Exit main routine	206
6.2	Steps for installing and modifying the exit code sample	206
6.3	Test for scenario 1	208
Chapter 7. PKI Services and the Cryptographic Coprocessor		219
7.1	Introduction to Cryptography Solution on S/390 - zSeries	220
7.1.1	Cryptographic Coprocessor Feature (CCF)	221
7.1.2	PCI Cryptographic Coprocessor (PCICC)	224
7.1.3	PCI Cryptographic Accelerator (PCICA)	226
7.1.4	Assigning coprocessors to an LPAR	227
7.2	Cryptographic solution on z990	228
7.2.1	CP Assist for Cryptographic Function	228
7.2.2	PCI Extended Cryptographic Coprocessor	228
7.2.3	Software requirements	230

7.3 Integrated Cryptographic Services Facility	230
7.3.1 CKDS and PKDS	232
7.3.2 Controlling access to ICSF resources	232
7.4 Boosting SSL connection with hardware encryption	232
7.4.1 Secure Sockets Layer (SSL)	232
7.4.2 IBM HTTP Server accessing the cryptographic coprocessor	234
7.4.3 Checking hardware encryption for Web server encryption	234
7.5 Keeping your CA signature key secure with ICSF	235
7.5.1 RACF taking advantage of ICSF	235
7.6 Sharing PKDS in a sysplex environment	237
Chapter 8. LDAP enhancements for availability	239
8.1 Optional LDAP enhancements for availability	240
8.1.1 Redundancy	240
Appendix A. PKI Exit sample	249
Appendix B. List of sample files provided with PKI Services	265
httpd.conf sample for PKI Web server 1	266
httpd.envvars sample for the PKI Web server	267
httpd.conf sample for PKI Web server 2	267
pkiserv.conf	268
pkiserv.envvars	271
pkiserv.tmpl	272
PKI Services subcomponents and message levels	330
JCL samples	330
Related publications	337
IBM Redbooks	337
Other publications	337
Online resources	338
How to get IBM Redbooks	338
Index	339

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

CICS®

DB2®

@server®

IBM®

Lotus Notes®

Lotus®

Multiprise®

MQSeries®

MVS™


Notes®

OS/390®

Parallel Sysplex®

RACF®

Redbooks™

Redbooks (logo) ™

RMF™

S/390®

WebSphere®

World Registry™

z/OS®

zSeries®

The following terms are trademarks of other companies:

Intel is a trademark of Intel Corporation in the United States, other countries, or both.

Microsoft is a trademark of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Preface

This IBM® Redbook is designed to help you understand and implement PKI (public key infrastructure) Services on z/OS®. As an overview of PKI Services on z/OS, this book looks into two different ways of installing PKI Services: the traditional step-by-step configuration that you do yourself, and by using the new IKYSETUP configuration utility, which enables you to enter certain parameters and then run IKEYSETUP to configure PKI Services for you.

We also look at the PKI Exit and show an example. A chapter about PKI Services and the Cryptographic Coprocessor discusses the Cryptographic solution on the IBM @server zSeries 990 and shows how to boost your SSL connection with hardware encryption.

The team that wrote this redbook

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Chris Rayns is the Project Leader and an IT Specialist at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on all areas of S/390® Security. Before joining the ITSO, Chris worked in IBM Global Services in the UK as a IT Specialist.

Roland Trauner is an IBM certified professional, responsible for WebSphere® Brand Management zSeries in Germany, Austria, and Switzerland. He worked as an IT consultant at the International Technical Support Organization, Poughkeepsie Center, for three years and was responsible for OS/390® e-business, network computing, and WebSphere.

Theo Antoff is a Senior RACF® Architect in Australia and director of his own companies in Australia (AG Glomar Pty Ltd.) and USA (Antoff IT, Inc.), consulting on all aspects of IBM Security Server components. He has 15 years of experience in RACF and MVS™ Systems programming. His projects include SDSF, CICS®, and DB2® conversions to RACF, CA-Top Secret to RACF migrations, merging RACF databases, and Security Technical Reviews. In his previous career as a physicist, he was involved in the research and development of the technology of non-volatile memories based on MIS structures. He has worked for IBM in Australia for three years.

Vicente Ranieri Jr. is a Senior Certified Consulting I/T Specialist in Brazil supporting customers across Latin America. He has 22 years of experience working with IBM customers, 13 of which were spent providing technical support for MVS, OS/390, and zSeries S/390 platforms. His areas of expertise include the zSeries platform, z/OS, Parallel Sysplex®, and zSeries Cryptographic solutions. He holds a postgraduate degree in Marketing Administration from Fundacao Getulio Vargas (EAESP-FGV), Sao Paulo.

Jack Jones is a Certified I/T Specialist working for the S/390 New Technology Center in Poughkeepsie, NY. He has 23 years of experience in the MVS and OS/390 data processing fields. His areas of expertise include system programming, data management, and security. Jack's current area of work is OS/390 for e-business, working with customers to ready their OS/390 systems for Internet access. Jack is a regular presenter at international conferences such as Guide, SHARE, the International Security Conference, and OS/390 Expo.

Patrick Kappeler is a former computer specialist in the French Air Force who joined IBM in 1970 as a diagnostic programs designer. He has held several specialist and management positions as well as international assignments, all dealing with S/390 technical support. He joined the EMEA S/390 New Technology Center in Montpellier in 1996, where he now provides consulting and pre-sale technical support in the area of e-business security.

Thanks to the following people for their contributions to this project:

Richard Conway
International Technical Support Organization, Poughkeepsie Center

Jim Sweeny
IBM z/OS Security Server design and development, Poughkeepsie

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions while getting hands-on experience with leading-edge technologies. Residents team up with IBM technical professionals, Business Partners, and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us.

We want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- ▶ Use the online **Contact us** review redbook form found at:

ibm.com/redbooks

- ▶ Send your comments in an e-mail to:

redbook@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Security Server PKI Services

This chapter discusses the Security Server PKI Services. We cover the following areas:

- ▶ Overview of digital certificates
- ▶ PKIX standards
- ▶ z/OS PKI Services
- ▶ Security Server PKI Server enhancements in z/OS V1R4

1.1 Overview of digital certificate

When compared to other known means of strong authentication, digital certificates (and the underlying public key cryptography algorithm) appear to be probably the best solution to the current authentication and encryption problem involving a very large population of users over a non-secure network such as the Internet.

However, the use of digital certificates, over the Internet or in an intranet environment, requires a supporting Public Key Infrastructure (PKI), which is the set of services, tools, and policies that enable the use of public key cryptography and management of keys and certificates in a PKI domain. The certificates and the associated key pairs are expected to have a life cycle as described in Figure 1-1 and Figure 1-2 on page 6.

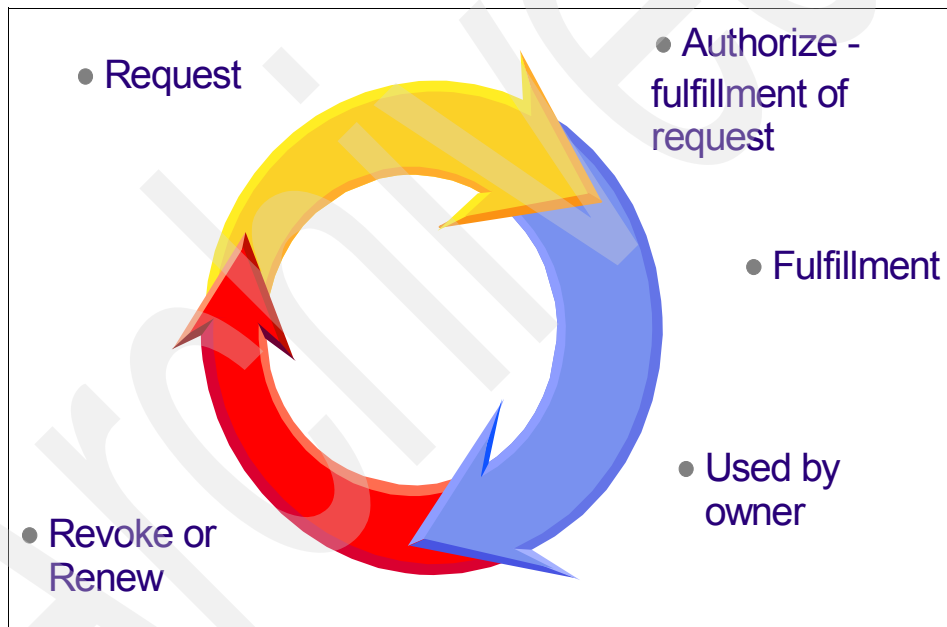


Figure 1-1 The digital certificate life cycle

The life cycle stages, in order, are:

1. Request phase

First there is the generation of an asymmetric algorithm key pair. Usually this generation is performed locally by the entity that requests the certificate, although some PKI implementations can perform the key generation on behalf of the certificate requestor. In that case, the PKI has to be able to deliver the generated private key securely to the certificate's owner.

The generation of a certificate request usually contains:

- The distinct name of the requestor
- The value of the public key
- Miscellaneous additional fields

This is all signed by the requestor's private key as a proof of origin of the certificate request.

2. Authorize Fulfillment of Request and Fulfillment phases

Generation and signature of the certificate itself by a Certification Authority (CA). The CA, which can be assisted by a Registration Authority (RA) in charge of verifying the validity and integrity of the certificate request and approving it, is the pivotal entity in a PKI. The CA is in charge of digitally signing the certificate (that is, vouching for the binding of the public key value to the name of the certificate's owner), thus making the certificate usable for strong authentication or other cryptographic processes. Strictly speaking, the RA provides for the *Authorize fulfillment of request* phase, and the CA performs the *Fulfillment*.

These two phases heavily engage the responsibility, and in many cases the liability, of the CA, which is to use the proper administrative procedures and highly secure technologies to ensure the integrity of its digital signature and of the signed contents.

Note that a certificate, as delivered by a CA, is granted a validity period determined by the CA policy. Usually, an end user certificate is valid for one year.

3. Used by Owner phase

The certificate can now be used by the owner for the purpose of authentication, which works as long as the requestor can demonstrate possession of the corresponding private key or any other cryptographic process where the value of one's public key is required. Note that the recipient of a certificate must have the public key of the CA that signed this certificate. This public key itself is delivered in a CA's certificate because, by the PKI principle, a CA is the only entity delivering certificates in a PKI domain.

4. Revocation or Renewal phase

The validity of a certificate can be denied in two ways: either the validity period of the certificate is over or the certificate is part of a Certificate Revocation List (CRL), which is issued by the CA that initially issued the certificate, usually on request from the certificate's owner.

The owner may request the revocation of a certificate for many reasons, including:

- Change of name of the owner

- Change of association between the owner and the CA (for example, when an employee leaves a company that is its own CA)
- Compromise or suspected compromise of the corresponding private key

The entity receiving a certificate checks for its expiration based on the receiver's local time and date. Verifying that a certificate is not part of a Certificate Revocation List requires the receiving entity to fetch the CRL from its repository, which usually is an LDAP directory (although some PKI implementations provide access to a CRL via the HTTP protocol). A certificate becomes part of a CRL at the completion of the *Revocation* phase.

During its normal life cycle, a certificate is set to expire after a certain validity period that is indicated in the certificate at its creation. The supporting PKI must then provide a way to renew the certificate, keeping the same certificate but with a new validity period and a new certificate serial number. This is the *Renewal* phase.

1.2 The PKIX standards

The digital certificate life cycle as described above is well documented and has been well understood in the industry since the late 1980s. However, the design and implementation of supporting technologies proved to vary from vendor to vendor. One of the major problem encountered when attempting to implement a PKI was the lack of interoperability between PKI products, hence the lack of potential scalability or capability to aggregate different PKI domains. This proved to be a severe impediment to the development of PKI use in the industry.

To address this issue, the Internet Engineering Task Force (IETF) launched the Public-Key Infrastructure (X.509) working group (PKIX) in 1995. Details of the group's achievements can be found at:

<http://www.ietf.org/html.charters/pkix-charter.html>

The PKIX working group compiled this set of definitions:

Certification Authority (CA)

An authority trusted by one or more users to create and assign public key certificates. Optionally, the CA may create the user's keys. Note that the CA is responsible for public key certificates during their lifetime, not just for issuing them, meaning that the CA also provides certificate revocation information.

Public Key Certificate (PKC)

A data structure containing the public key of an end entity, and other information, which is digitally signed with the

private key of the CA that issued it. PKC should not be confused with other types of certificates that the PKIX group is working on, such as the Attribute Certificate (AC). In this book we explicitly refer to the PKC, which is the X.509 V3 certificate.

End Entity (EE) User of PKI certificates or the end user system that is the subject of a certificate.

Subject The entity named in a Public Key certificate. Subjects can be human users, computers (represented by Domain Name Service [DNS] names or Internet Protocol [IP] addresses), or even software agents. The subject is often referred to as the *owner* of the certificate.

Registration Authority (RA) An optional entity associated with the CA. Responsible for performing some of the administrative tasks necessary for registering subjects, such as: confirming the subject's identity, validating that the subject is entitled to have the values requested in a Public Key Certificate, and verifying that the subject has possession of the private key associated with the public key in the PKC request.

Public Key Infrastructure (PKI) The set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke PKCs based on public-key cryptography.

Certificate Policy (CP) A named set of rules that indicates the applicability of a public key certificate to a particular community or class of application with common security requirements. For example, a particular certificate policy might indicate applicability of a type of public key certificate to the authentication of electronic data interchange transactions for the trading of goods within a given price range.

Certification Practice Statement (CPS) A statement of the practices that a CA employs in issuing public key certificates.

Top CA, or root CA A CA that is at the top of a PKI hierarchy.

The ITU-T Recommendation X.509 has been accepted as the basis for PKIX PKI, both as data formats and procedures related to distribution of public keys via PKCs digitally signed by CAs. X.509 evolved to specify fields in addition to the initial certificate format, so that today PKIX-compliant PKIs use the X.509 V3 certificate format shown in Figure 1-5 on page 12. Recommendation X.509 also

describes a method for publishing certificate revocation information that involves each CA periodically issuing a signed CRL, which is a list that identifies the references of revoked certificates. The current PKIX CRL is at the X.509 V2 format and is shown in Figure 1-6 on page 13.

A simplified graphical view of a PKIX-compliant PKI is shown in Figure 1-2, not fully representing all of the items addressed by the PKIX Working Group but only the ones relevant to this book:

- ▶ The PKCS#10 format for the certificate request
- ▶ The X.509 V3 format for the signed certificate
- ▶ The X.509 V2 format for the CRL
- ▶ The LDAP protocol to reach the CRL residing in an LDAP directory
- ▶ The OCSP protocol for real-time access to revocation information

For more information, visit:

<http://www.ietf.org>

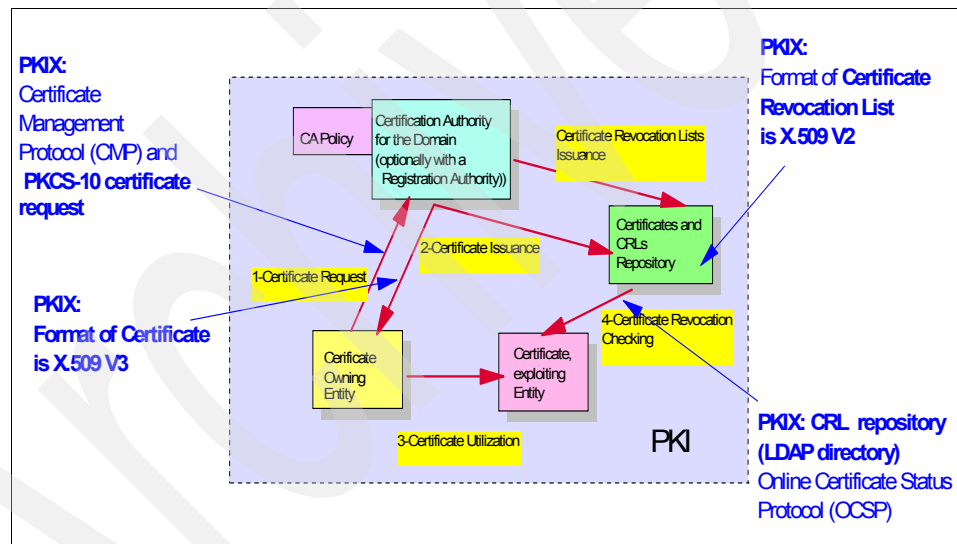


Figure 1-2 A typical PKIX PKI

1.2.1 CA hierarchy

For various reasons, certification authorities may be organized in a hierarchy, as shown in Figure 1-3 on page 8, in which security domains B and C are each

covered by their own Certification Authority and are part of a hierarchy with Certification Authority A at the top. In more practical terms:

- ▶ CA B and C are intermediate CAs. They have an asymmetric key pair, using their private key to sign certificates to be used by end entities in domain A or B. However, their own certificates are not self-signed, but instead signed by the top CA (A). Only CA A has a self-signed certificate.
- ▶ As a consequence, verifying a certificate signed by either B or C requires also verifying that the certificate of either B or C is properly signed by A. This is a chain of certificates, or a *certification path*.
- ▶ Therefore, in a CA hierarchy or certification path, you can expect that the top (root) CA's certificate is the only one to be trusted. An example of a trusted CA is a CA with a self-signed certificate: You simply have to trust it because nobody else vouches for it.
- ▶ Protocols using digital certificates, such as SSL/TLS, take care of providing the certificate chain, if any, to a certificate recipient. Each certificate provided in the chain is used on the fly to verify the certificate just below it, up to the point where the chain presents a certificate signed by the root CA. This will be the last verification performed because, as previously stated, the root CA certificate must be trusted by the certificate recipient.

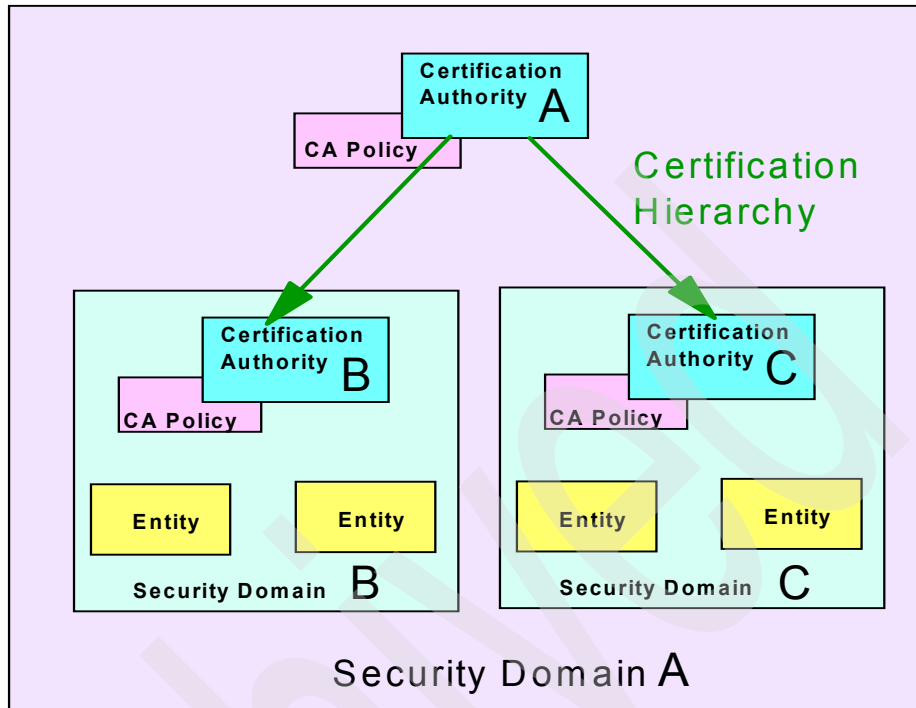


Figure 1-3 CA hierarchy

Trust transitivity with cross-certification

Trust can be implemented between different security domains by using the cross-certification of CA certificates. Cross-certification is the act of providing a CA with a certificate signed by another CA in another hierarchy. Figure 1-4 on page 9 is an example of cross-certification, in which domains B and F have been consolidated into one trust domain. CA B is cross-certified by CA F. An entity in domain B can provide a certificate chain comprising the cross-certification certificate to an entity in domain F. Because this certificate is signed by F, the certification path will be verified and the entity in B will be trusted by the certificate recipient in F.

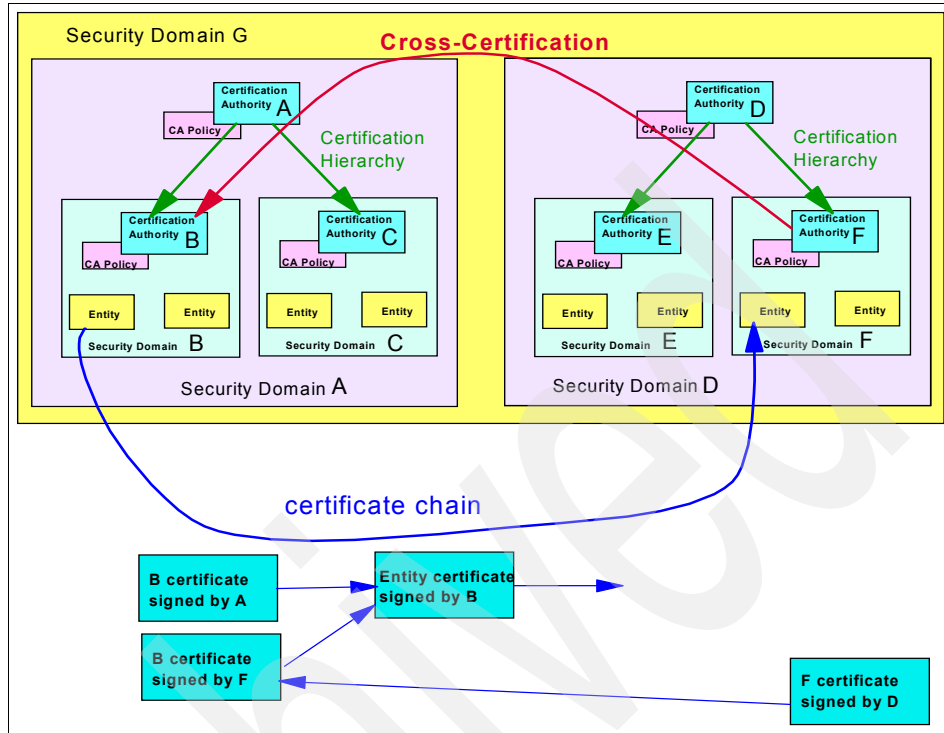


Figure 1-4 Cross-certification of Certification Authorities

1.2.2 The X.509 certificate and Certificate Revocation List

The PKIX documents emphasize the following basic principles when it comes to using a digital certificate:

- ▶ Recipients of digital certificates must be confident that any time they rely on a public key, the subject that they are communicating with owns the associated private key. This applies whether an encryption or digital signature mechanism is used. This confidence is obtained through the use of protocols (for example, SSL/TLS) that cannot carry forward the communication if this condition is not fulfilled.
- ▶ A certificate has a limited valid lifetime, which is indicated in its signed contents. Because a PKC's signature and timeliness can be checked independently by a certificate-using client, certificates can be distributed via untrusted communications and server systems, and can be cached in unsecured storage in certificate-using systems.

- ▶ Certificates are used in the process of validating signed data or securely transmitting encryption keys. Specifics vary according to which algorithm is used, but the general process works as follows:
 - a. The recipient of signed data verifies that the claimed identity of the user is in accordance with the identity contained in the certificate.
 - b. The recipient validates that no certificate in the path is revoked (for example, by retrieving a suitably current CRL or querying an online certificate status responder) and that all certificates are within their validity periods at the time the data was signed.
 - c. The recipient verifies that the data is not claimed to have any values for which the certificate indicates that the signer is not authorized.
 - d. The recipient verifies that the data has not been altered since signing, by using the public key in the certificate.

If all of these checks pass, the recipient can accept that the data was signed by the purported signer. As these basic principles always stand true, it appeared also during practical experimentation of digital certificates that they had to be supported by more sophisticated structures than initially planned. Actually, the X.509 digital certificate format had to go through evolutions as explained in the following paragraphs.

X.509 Certificate Version 1

This was the initial version of 1988. The specified fields, which are still in use today in the Version 3 format, are:

- ▶ Version number (1).
- ▶ Serial number: the serial number assigned by the CA at signature time.
- ▶ Signature algorithm ID: an indication from the CA of which algorithm has been used to signed the certificate.
- ▶ Issuer's name: the name of the certification authority. The name appears as a *distinguished name* in the X.500 syntax.
- ▶ Validity period: the CA imposes a limited validity period to the certificate, usually one year from its signing date.
- ▶ Subject's name: The owner of the public key, hence owner of the certificate as well. The name appears as a *distinguished name* in the X.500 syntax.
- ▶ Subject's public key information: The numeric, non-secret value of the certificate owner's public key, along with information about the algorithm this key is intended for.

Note: It takes only two pieces of information to uniquely identify a digital certificate: the Certification Authority's (Issuer's) name and the serial number assigned to the certificate by the CA.

X.509 Certificate Version 2 (1993)

The X.509 Version 2 certificate added two new fields to accommodate the fact that no one can really expect that in the real world x.500 distinguished names will ever be unique:

- ▶ Issuer Unique Identifier: This optional field enables specifying a complementary unique identifier to the Certification Authority in case the Issuer's distinguished name could already be in use by another entity.
- ▶ Subject Unique Identifier: This optional field enables specifying a complementary unique identifier to the certificate owner in case the Subject's distinguished name could be in use already by another entity.

X.509 Certificate Version 3 (1995)

Still facing real-life facts, it appeared that new certificate fields were needed that could be implemented and used as needed by specific applications. These extension fields appeared in Version 3, yielding a certificate format as shown in Figure 1-5 on page 12. The intent of Version 3 was to address some of the security concerns and limited flexibility from Version 1 and 2.

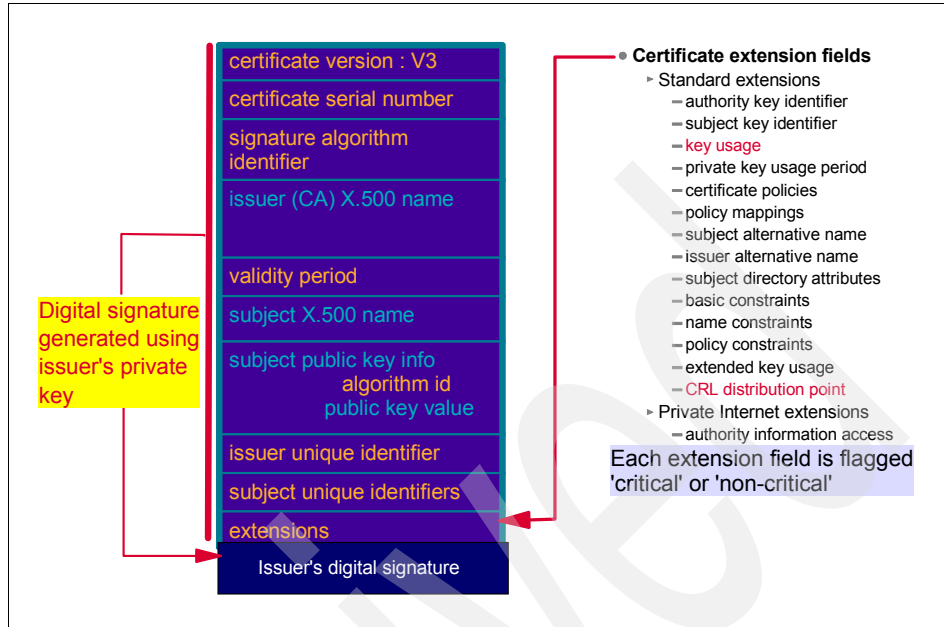


Figure 1-5 X.509 V3 Certificate

X.509 Certificate Revocation List

The Certificate Revocation List went under its own evolution. It is now at Version 2, and it contains the fields shown in Figure 1-6 on page 13. Note that a CRL is intended to be stored in an LDAP directory, from which it can be fetched, and, as the CRL is signed by the issuing certification authority, the CRL repository is not required to be secure. A CA uses the same repository to make new certificates available and to publish CRLs.

More about the Certificate Revocation List

- ▶ The CRL is intended to be published; that is, it will be updated regularly in the LDAP directory by the CA. It is up to the CA to indicate the frequency of these updates in its CA revocation policy.
- ▶ The implication of these periodic updates is that the revoked status of a certificate will be known from potential recipients only after the CRL update that follows the revocation has been issued, which may be several hours or days after the revocation request has been successfully processed. To help circumvent this timeliness problem, another protocol has been proposed by IETF: the Online Certificate Status Protocol (OCSP), through which the revocation status of a certificate can be obtained in real time. However, the use of the CRL (instead of OCSP) prevails in PKIs.

- The CRL can be located in the LDAP directory:
 - As an attribute in the CA entry (that is, the entry with the CA's distinguished name). This is known as the *global CRL*. The global CRL is added any new revoked certificate information, whereas previously revoked certificate can be deleted, if appropriate, from the list at each update. For large PKI domains, the global CRL can grow very large.
 - As a directory entry by itself, called a *distribution point*. The distribution point approach is intended to break down the complete CRL into smaller pieces, each one being a separate directory entry easier to access than the whole global CRL. The distinguished name of the distribution point entry to look into is indicated in an extension field of the published certificate. (See 1.2.3, "The x.509 v3 certificate extension fields" on page 14 for an explanation of extension fields.) The distribution point entries are leaf nodes in the directory below the CA entry.

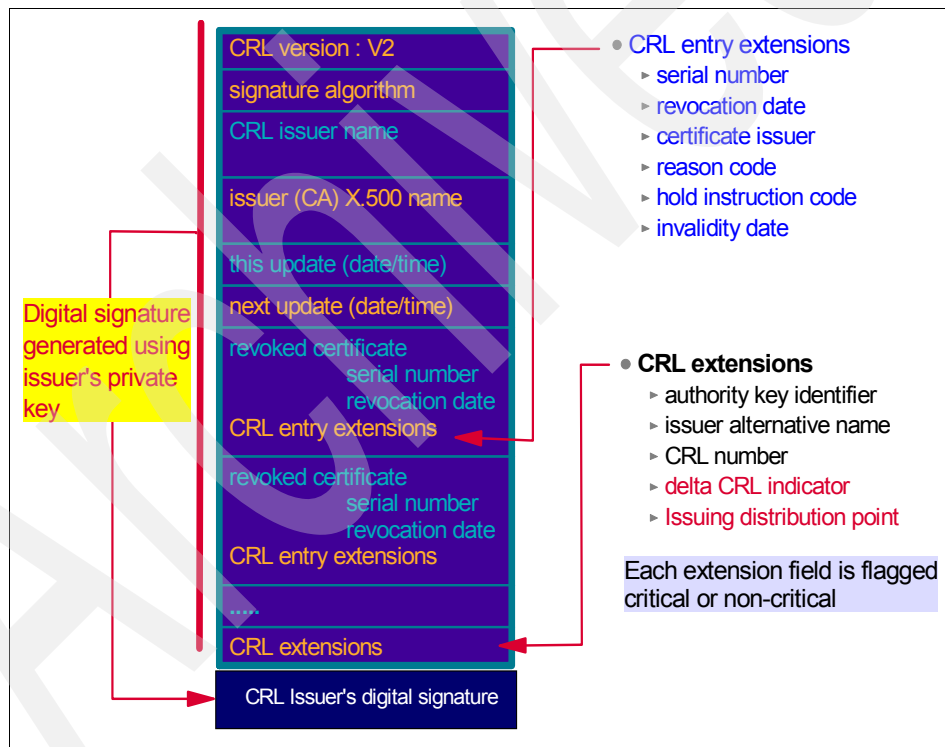


Figure 1-6 X.509 V2 Certificate Revocation List

1.2.3 The x.509 v3 certificate extension fields

The extension fields are optional fields of which creation, name, and contents are just a matter of convention established with the certificate-exploiting application. An extension is agreed upon in the context of the use of the certificate by a specific application, and it is given a name, a value, and a criticality flag.

The idea of the criticality flag is to state how important it is for the certificate recipient application to recognize the content of the field. If the certificate exploiting application is not able to handle an extension field, then the process is carried on if the extension is marked as non-critical. It is aborted if the extension is marked as critical. Not recognizing an extension marked as critical can potentially lead to severe operating or security exposures.

In consequence, the structure of a certificate extension consists of the following fields:

Type	Specifies the format of the value field
Criticality	A one-bit flag, used as explained above
Value	The actual extension data

Although the extensions are intended to provide whatever flexibility and meaning an application requires from the contents of the certificate, experience showed that the use of certificate extensions by different real applications were converging in terms of needs and meanings. Therefore, for the sake of interoperability via extension fields standardization, X.503 also specifies the contents of four sets of standard extension fields, which are:

- ▶ Extensions with information pertaining to the public key
- ▶ Extensions with information pertaining to the use of the certificate
- ▶ Extensions specifying attributes for the owner and the certification authority
- ▶ Extensions with information about the certificate hierarchy

These fields are described in RFC 3280. Next, we provide a brief overview of their purpose and indicate the recommended settings of the criticality flag.

Extensions with information pertaining to the public key

This section looks at extensions that contain information specifically about the public key.

Authority key identifier (non-critical)

This field indicates the unique identity of the key used by the CA to sign the certificate. This covers the case where the CA has used several signature keys since it was put into operation. The field value is intended to speed up the retrieval process of the correct CA public key by the certificate receiving application.

Subject key identifier (non-critical)

This field indicates the unique identity of the public key enclosed in the certificate. This covers the case where the certificate owner has had, or will have, several keys. It is intended to speed up the retrieval process of the correct private key.

Key usage (critical)

This field specifies the utilization domain of the public key in the certificate. The standardized usages are:

Digital signature	Any signature other than keyCertSign or cRLSign.
Non Repudiation	The subject's public key can be used by a non-repudiation service to verify signature.
keyEncipherment	The subject's public key is used for secure key transport.
dataEncipherment	The subject's public key is used for data confidentiality.
keyAgreement	The subject's public key is used by a key agreement protocol, such as Diffie-Hellman. It must go along with either the encipherOnly or decipherOnly bit.
keyCertSign	The subject's public key is used to verify a signature on a digital certificate.
cRLSign	The subject's public key is used to verify a signature on a certificate revocation list.
encipherOnly	The subject's public key is used by a key agreement protocol, and it can only be used for enciphering data while performing key agreement.
decipherOnly	The subject's public key is used by a key agreement protocol, and it can only be used for deciphering data while performing key agreement.

CRL distribution point (non-critical)

This extension provides the location, usually an LDAP URI, of the Certificate Revocation List to check for the certificate's validity.

Freshest CRL (non-critical)

Also known as *Delta CRL distribution point*. It identifies, in a syntax similar to the CRL Distribution Point, how delta CRL information is obtained.

Extended key usage (critical or non-critical)

This extension indicates that the subject's public key may be used for purposes other than what is indicated in the key usage extension:

serverAuth	Used for TLS server authentication.
-------------------	-------------------------------------

clientAuth	Used for TLS client authentication.
codeSigning	Used to sign downloadable code.
emailProtection	Used to protect e-mail.
timeStamping	Used for binding the hash of an object to a time.
OCSP signing	Used to sign OCSP responses.

Extensions with information pertaining to certificate use

This section looks at extensions that contain information specifically about certificate use.

Certificate policies (critical or non-critical)

This extension contains information about the CA policy under which the certificate has been issued and the purposes for which the certificate may be used. This information is provided as an OID (Object Identifier) number. (See “The Object Identifier (OID)” on page 18.) If the extension is marked critical, then the exploiting application must abide with the designated policy.

Policies Mappings (non-critical)

This extension is used in intermediate CA certificates. It establishes correspondence as designed by the issuing CA. An exploiting application can therefore make a decision about accepting a subject CA policy based on the mapping indicated by the issuing CA.

Extensions with information pertaining to certificate owner and certification authority attributes

This section looks at extensions that contain information specifically about the certificate owner and certification authority attributes.

Subject alternative name (critical if certificate subject field is empty)

This extension enables additional identities to be bound to the subject of the certificate. That is, the subject can be designated by alternate identities such as:

- ▶ An Internet e-mail address
- ▶ A DNS name
- ▶ An IP address
- ▶ A uniform resource identifier (URI)

Several alternate identities can be specified in the extension.

Issuer alternative name (non-critical)

This extension enables use of Internet-style identities for the certificate issuer, as with the subject alternative name.

Subject Directory Attributes (non-critical)

This extension contains attributes and their values related to the identity of the subject. A typical identification attribute in this context is the subject's nationality.

Extensions with information pertaining to the certificate hierarchy

This section looks at extensions that contain information specifically about the certificate hierarchy.

Basic constraints (critical)

This field indicates whether the subject is an end entity or a CA. If the subject is a CA, it gives the certification path depth (that is, it gives the maximum number of non-self-issued intermediate certificates that may follow this certificate in a valid certification path).

Name constraints (critical)

The name constraints extension is used only in a CA certificate. It indicates a name space within which all subject names in subsequent certificates in a certification path must be located. It is intended to specify acceptable cross-certification domains.

Policy constraints (critical or non-critical)

The policy constraints extension can be used in certificates issued to intermediate CAs. It can be used to prohibit policy mapping or require that each certificate in a path contain an acceptable policy identifier.

InhibitAnyPolicyConstraint (critical)

This extension can be used in certificates issued to intermediate CAs. "Inhibit any policy" indicates that the special anyPolicy identifier is not considered an explicit match for other certificate policies.

1.2.4 Certificate and CRL appearance

A digital certificate, by nature, is intended to be public knowledge. That is, the enclosed information appears unencrypted and the only piece of encrypted data is actually the CA's digital signature (of which content does not bring any additional functional information). A simplified graphical view of a certificate and its contents is shown in Figure 1-7 on page 19, where fields contain numeric or alphanumeric values.

However, for interoperability reasons, the contents of a certificate must go through specific syntax and encoding (not encryption) transformations that render a certificate readable by software only.

Figure 1-8 on page 20 shows a simplified overview of these transformations.

It begins with abstract notation

As part of the OSI effort in the late 1980s, a need was recognized for an abstract description language of data structures that could be translated easily into a high-level language for software manipulation of data, but would remain platform-neutral for the sake of interoperability. Abstract Syntax Notation 1 (ASN.1) was created and has since been used extensively whenever it came to formulate how data (such as RFCs) should be structured to be processed in highly heterogeneous environments such as the Internet. ASN.1 is further described in ITU-T recommendation X.208.

An example of the ASN.1 syntax is shown in Example 1-1. It is the description of the issuer's (CA's) name as it should appear in an X.509 certificate.

Example 1-1 An example of ASN.1

```
Name ::= CHOICE {
    RDNSSequence }
RDNSSequence ::= SEQUENCE OF RelativeDistinguishedName
RelativeDistinguishedName ::=
    SET OF AttributeTypeAndValue
AttributeTypeAndValue ::= SEQUENCE {
    type      AttributeType,
    value     AttributeValue }
AttributeType ::= OBJECT IDENTIFIER
AttributeValue ::= ANY DEFINED BY AttributeType
DirectoryString ::= CHOICE {
    teletexString      TeletexString (SIZE (1..MAX)),
    printableString    PrintableString (SIZE (1..MAX)),
    universalString    UniversalString (SIZE (1..MAX)),
    utf8String         UTF8String (SIZE (1..MAX)),
    bmpString          BMPString (SIZE (1..MAX)) }
```

The Object Identifier (OID)

One of the ideas that accompanies ASN.1 is that of designating objects (such as cryptographic algorithms, data types, customized sets of information such as policies, and so on) that are meaningful for whoever will exploit the data structure. Instead of providing a name for the object, an OID (Object Identifier) is given instead.

The OID is a unique set of numbers that designate the object. Because OIDs are used by millions and must be as unique as the objects they represent, they are distributed by a central registry organization, the IANA (Internet Assigned Numbers Authority, <http://www.iana.org>). The OID numbering space is federated between all using organizations. ISO specifies what numbers in the OID refer to which organization in a hierarchical manner. A subtree created in the hierarchy for an organization is called an *arc*.

For example, if the keyusage extension in an X.509 certificate is designated by OID 2 5 29 15, this can be broken down as:

- 2** The joint ISO-ITU numbering space
- 5** Part of the X.500 directory arc
- 29** Certificate extension
- 15** Keyusage

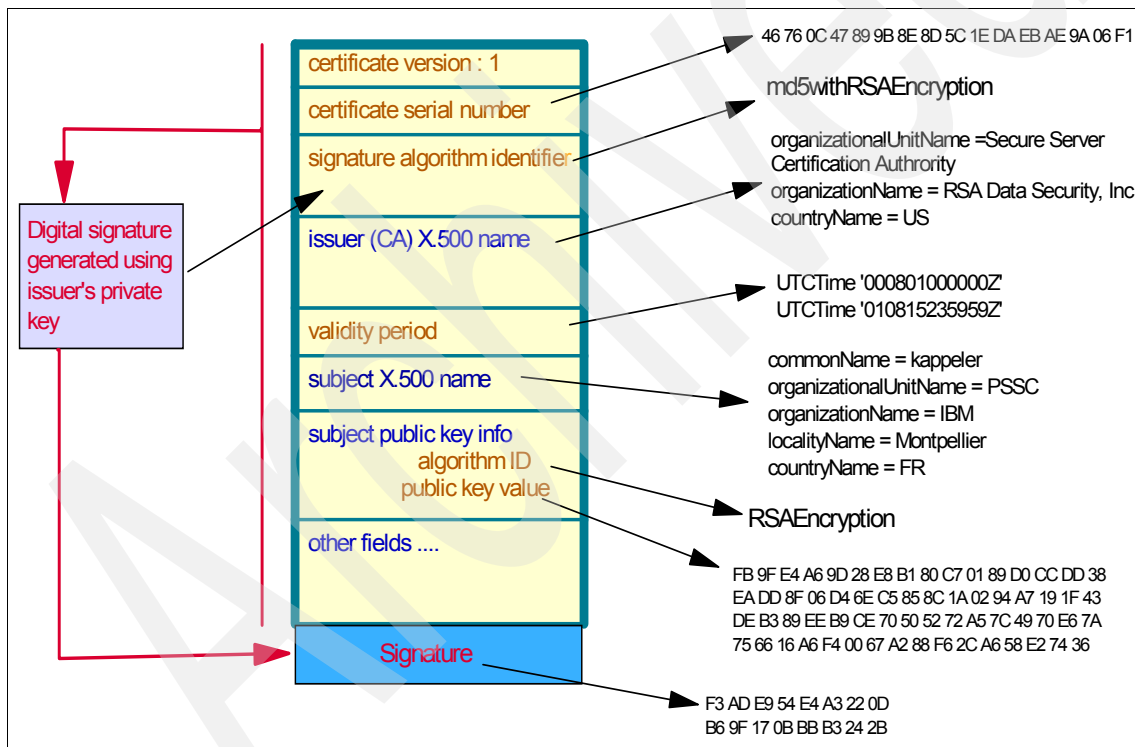


Figure 1-7 Contents of a digital certificate

The BER and DER encoding rules

After the data has been described using ASN.1, it must be translated into bit strings to be sent to other computers. ASN.1 comes with a set of binary encoding rules: the Basic Encoding Rules (BER) and the Distinguished Encoding Rules

(DER, which is actually a subset of BER). These encoding rules introduce proper length and data type information in the binary string that represents an object and its value, to be exchanged between computers.

For various reasons not discussed here, DER is the most preferred encoding method for values that will be digitally signed. DER is defined in section 8.7 of the CCITT Recommendation X.509.

As far as we are concerned here, the net result is that what originally was ASCII characters on octet boundaries is now part of the resulting binary strings and does not appear as the original value any more. Hence the strange look of the certificate contents when attempting to display it as in Figure 1-8.

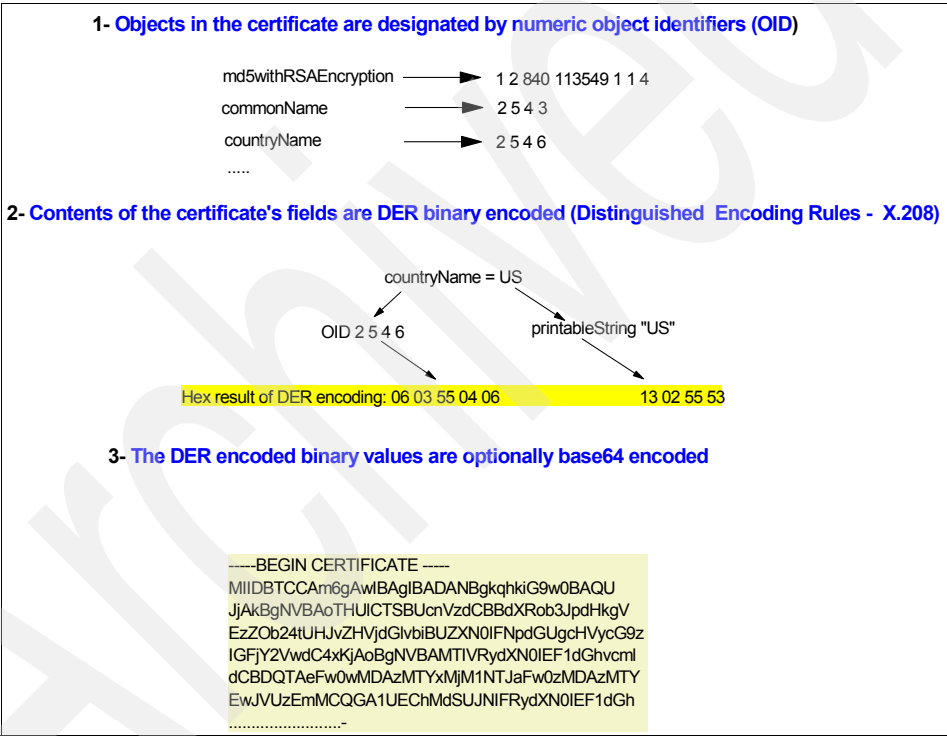


Figure 1-8 “On the wire” syntax and encoding of digital certificates

The optional base-64 encoding

The DER format of a certificate can lead to problems with some transmission devices, as they may wrongly recognize some bit configurations in the binary string as special or control characters intended to influence their operations.

If there is such an exposure, the DER format is being encoded (again: not encrypted) using the base-64 algorithm. The binary input is transformed by the algorithm into a chain of printable ASCII characters, as opposed to unknown or control characters, which can then be transmitted without any potential problems. The base-64 encoding algorithm is reversible. It is widely documented in numerous Web sites.

1.3 The z/OS PKI Services

This section looks at the z/OS PKI Services, reviewing the prerequisite products required for installation, the requests supported by the z/OS PKI Services, and giving an overview of the z/OS PKI Services Architecture.

1.3.1 Security Server PKI Services in z/OS

PKI Services is a new component of the z/OS Security Server (beginning with z/OS V1R3). It is a complete Certification Authority package, always enabled independently of RACF but closely related from the functional standpoint to the SAF callable services interface, front-ended by RACF or an equivalent security product. The Certification Authority keys are located in an RACF (or equivalent) key ring. The z/OS PKI can be a root CA or an intermediate CA.

It provides these functions to implement and perform full certificate life cycle management:

- ▶ User request driven via customizable Web pages
- ▶ Automatic or administrator approval process
- ▶ End user / administrator revocation process

The user Web page interface is based on the PKISERV RACF SPE delivered in the OS/390 V2.10 time frame (downloadable from the RACF Web site). But whereas PKIServ is provided with a subset of PKI functions intended to be used for testing purposes or internally to an organization, the z/OS PKI Services offer the functions and the capacity required by large-scale external PKIs. The SAF interface provides the existing limited Security Server (RACF) certificate support, and, in addition, the R_PKIServ SAF callable services, which are extended to provide additional functions that support the programmatic request of certificates and certificate management used by the Web user and Web administrator functions.

With PKI Services, z/OS installations have the capability to establish a PKI infrastructure and serve as a certificate authority for internal and external users. The issuance and administration of digital certificates and certificate revocation

lists are performed in accordance with the CA policy that the owning organization puts in place.

The z/OS PKI Services is PKIX compliant and relies from the functional standpoint on the z/OS implementation of the Common Data Security Architecture (CDSA) from Intel® Corporation: the z/OS Open Cryptographic Services Facility (OCSF). CDSA supports multiple trust models, certificate formats, cryptographic algorithms, and certificate repositories. Therefore, it supports X.509 V3 certificates, and the z/OS PKI Services are primed to use the RSA algorithm for signature.

The user and administrative accesses to z/OS PKI Services is via HTTP/HTTPS only, both to request, revoke, or pick up certificates and to approve or reject requests by authorized administrators. Web pages delivered with z/OS PKI Services come with a set of customizable Web CGI scripts, and a programming exit is also included for advanced customization.

The created certificates or certificate revocation lists (CRL) are posted to an LDAP directory.

1.3.2 Prerequisite products

The following products must be installed prior to configuring PKI Services:

- ▶ IBM z/OS HTTP Server
 - IHS for z/OS comes in the base z/OS.
- ▶ LDAP directory
 - Although z/OS LDAP server is recommended, any V2-compliant LDAP server should fit (and LDAP V3 servers are expected to be downward compatible). The z/OS LDAP server comes in the base z/OS Security Server; however, the TDBM back end has to be used for storing CRLs and certificates, implying that DB2 also has to be running on the system.
 - A PKIX schema is required to be installed in the LDAP directory. The minimum schema shipped with z/OS LDAP server (schema.user.ldif) is PKIX compliant.

- ▶ Cryptographic Services

The following z/OS cryptographic services are required:

- Open Cryptographic Services Facility (OCSF). This is the z/OS implementation of the Intel Common Data Security Architecture (CDSA) cryptographic services. OCSF comes in the base z/OS Security Server.

- Open Cryptographic Enhanced plug-in (OCEP). OCEP enables using RACF as an OCSF (CDSA) data library and trust policy. OCEP comes in the base z/OS Security Server.

The following z/OS cryptographic service is optional:

- Integrated Cryptography Service Facility (ICSF) - ICSF drives the S/390-zSeries hardware cryptographic coprocessors and, if the coprocessors have been enabled in the system, can be used to offload the cryptographic workload from the software. In addition, hardware built-in security can be used to protect critical assets such as the CA private key.
- ▶ RACF (or equivalent)
 - As of the writing of this book, IBM does not know of equivalent functions provided by products intended to be a replacement for RACF.

1.3.3 Requests supported by z/OS PKI Services

As indicated before, the Web application provided with the z/OS PKI Services is highly customizable: Certificate requests from certain users can get automatic approval, extensions can be inserted into the certificates, the certificate validity period can be adjusted depending on the kind of certificate requested, and so on.

As delivered in z/OS, the PKI Services are primed to support the following certificate requests:

- ▶ One-year SAF server certificate. This certificate request is auto-approved but the requestor must provide a valid SAF user ID and password.
- ▶ One-year SAF browser certificate. This request is auto-approved but the requestor must provide a valid SAF user ID and password and a certificate label. A certificate image is stored in the RACF database, under the specified label, for further mapping between the certificate and the specified user ID.
- ▶ One-year PKI SSL browser certificate. The request has to be approved by an authorized administrator. The certificate is not stored in the RACF database, and mapping to a specific RACF user ID must be arranged by off-band means (for example, by the RACF admin or a z/OS self-registration Web application).
- ▶ One-year PKI S/MIME browser certificate. This is similar to the SSL browser certificate, with the addition of the AltEmail subject alternate name extension.
- ▶ Two-year PKI browser certificate for authenticating to z/OS. This certificate request is auto-approved but the requestor must provide a valid SAF user ID and password and a certificate label. The certificate is given the IBM HostIDMappings extension, with the specified RACF user ID value in order to establish certificate mapping in the recipient z/OS.

Note: The HostIDMappings extension (OID 1 3 18 0 2 18 1) is an IBM extension, also available for public use. RACF automatically maps a valid certificate to the RACF user ID provided in the extension. This mapping is controlled at the server level with the SERVAUTH profile class, and at the RACF policy level with the HIGHTRUST attribute in the CA DIGTCERT profile.

- ▶ Five-year PKI SSL Server certificate. There is no RACF user ID and password provided, so this certificate request has to be approved by an authorized administrator.
- ▶ Five-year PKI IPsec server certificate. This is a certificate similar to the SSL server certificate, with the adjunction of the keyusage extension with handshake and data encrypt encoded, and the subject alternate name extensions with AltEmail, AltIPAddr, AltURI, and AltDomain.
- ▶ Five-year PKI intermediate CA certificate. This certificate request is auto-approved but the requestor must provide a valid SAF user ID and password.

1.3.4 Browser and server certificates

A certificate is requested to and obtained from the z/OS PKI Services only via the HTTP/HTTPS protocol. This is a very straightforward process when obtaining a browser (client) certificate, but it is more complicated to request, obtain, and install a server certificate.

Browser (client) certificate process flow

Figure 1-9 on page 25 shows the typical process flow for browser certificates.

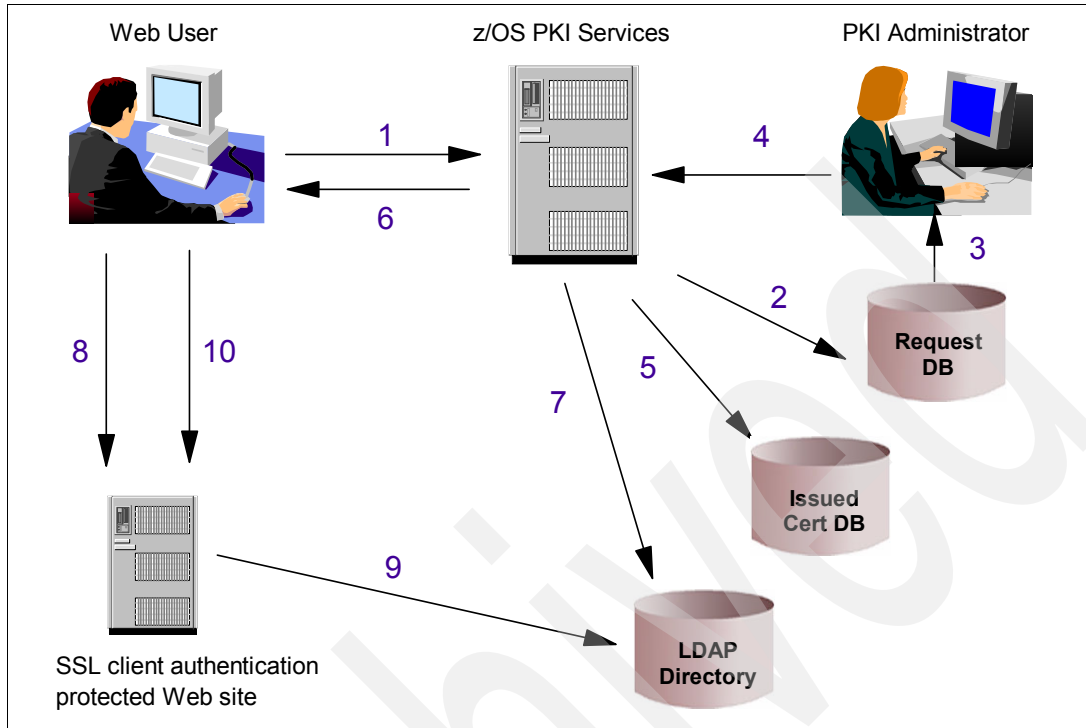


Figure 1-9 Browser certificate process

The process is:

- ▶ The Web user submits the PKCS#10 certificate request (1).
- ▶ The request is queued for approval by the administrator (2).
- ▶ The administrator reviews the request and approves or rejects it (3).
- ▶ If approved (4) it is issued and stored (5).
- ▶ The certificate is returned to the Web user when queried (6).
- ▶ It is also published to an LDAP directory (7). The certificate revocation list (CRL) is also published to LDAP on a continuous basis.
- ▶ The Web user uses the certificate to authenticate to an SSL client authentication protected Web site (8).
- ▶ The SSL handshake validates the certificate and checks the CRL (9).
- ▶ If everything is valid, the user gains access (10).

Server certificates process flow

Figure 1-10 shows the typical process flow for Web server certificates.

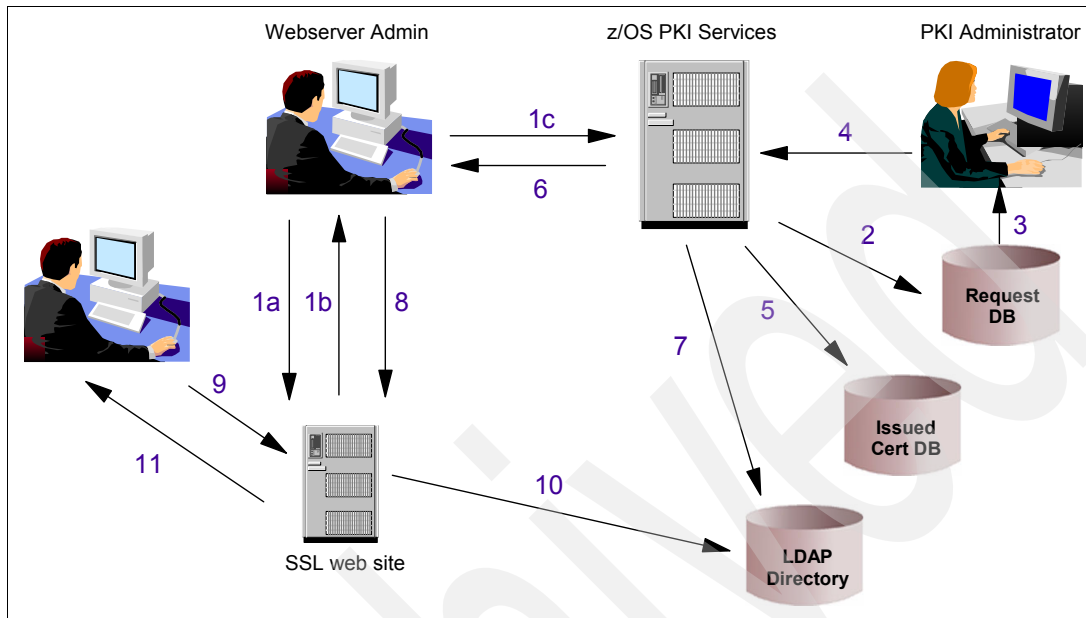


Figure 1-10 Server certificate process

The process is:

- ▶ The Web server administrator uses server-specific software to generate a PKCS#10 request (1a).
- ▶ This is copied (1b) and pasted (1c) into the certificate request Web page and submitted.
- ▶ Steps for queuing, approving, issuing, and retrieving the certificate are identical to the preceding browser flow (2-7).
- ▶ The Web server administrator installs the certificate into the Web server (8) and brings it online.
- ▶ Web users may now visit the SSL-protected Web site (9).
- ▶ If client authentication is enabled, the client's certificate is validated using the CRL in LDAP (10).
- ▶ If all is valid, the user gains access (11).

1.3.5 The z/OS PKI Services architecture

Figure 1-11 on page 27 illustrates the z/OS PKI Services architecture.

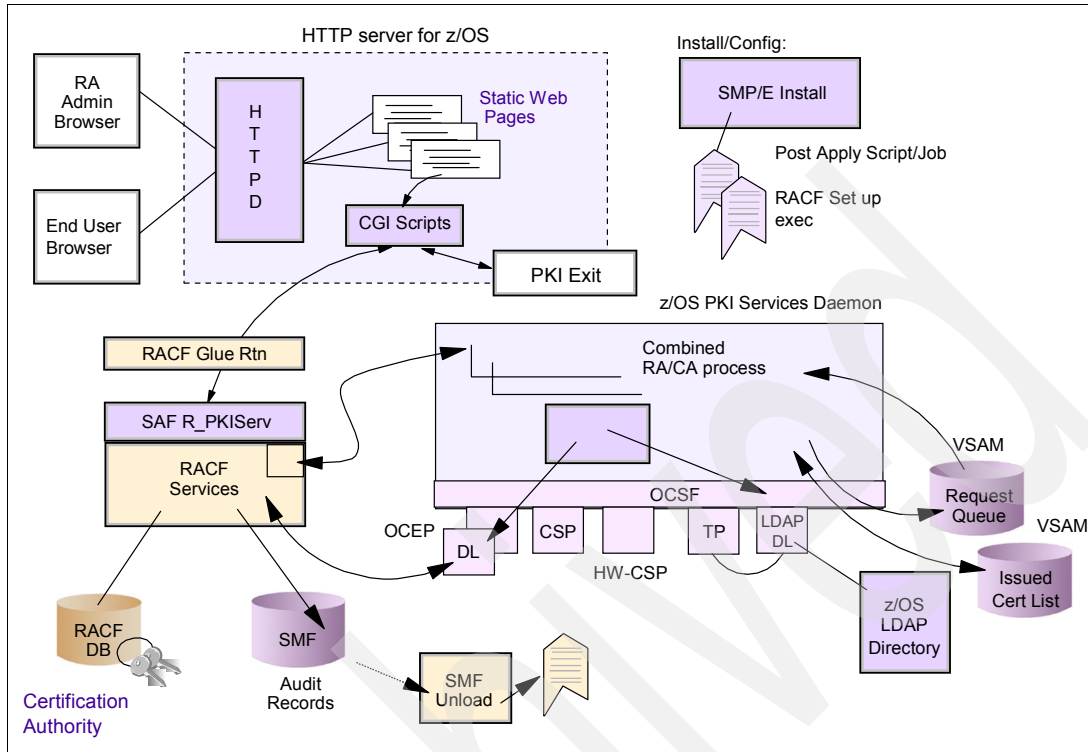


Figure 1-11 PKI Services architecture

The z/OS HTTP server provides the end-user and administrator interface. The customizable Web page makes use of CGI routines (also provided), and their contents are explicitly defined in a templates file that can be edited if necessary. The information in the template file is directly related to each type of request that the PKI supports, and it is exploited by the Web application (CGIs) to interface with the end user or the administrator and to submit requests to R_PKIServ SAF callable services interface when appropriate. CGIs are written in REXX, thus requiring an RACF glue routine because REXX cannot create the structure parameters required by the callable service. Optionally, the customer can use Pkiexit if more customization is needed.

The end-user administrative interface requires two instances of the z/OS Web server sharing the same certificate and private key:

- Instance one: supports HTTP and HTTPS for end users and PKI administrator. It is set up to prompt for SAF user ID and password depending on the requested URL.

- ▶ Instance two: supports only HTTPS with client authentication. It is required for authenticating a client certificate revocation request.

Any request, for end users and administrators, is always directed to instance one of the HTTP server first and may be automatically re-directed to instance two depending on what the request is.

R_PKIServ is a problem program service backed by RACF. Requests are verified by RACF and submitted to the PKI Services daemon, the core UNIX® application in the z/OS PKI Services. Additionally, RACF can create SMF auditing records.

End-user functions of the service are: request, retrieve, verify, revoke, and renew a certificate. Administrator functions of the service are: query, approve, modify, and reject certificate requests, and query and revoke issued certificates.

The PKI Services daemon is a multi-threaded server. It has service threads for incoming requests and background threads for housekeeping tasks such as the periodic issuance of a CRL and deletion of inactive requests. It maintains two VSAM data sets, one for storing all requests it receives (ObjectStore) and the other for keeping a list of issued certificates (ICL).

Figure 1-12 on page 29 focuses on the daemon and shows the configuration (pkiserv.conf) and environment variables (pkiserv.envars). These files are dedicated to the operations of the daemon itself, as opposed to the template file that directs the operations of the Web interface. The directives in pkiserv.conf drive such operational characteristics as:

- ▶ The number of initial service threads
- ▶ The RACF key ring where the CA keys are located
- ▶ The objects and their OIDs subject to customization
- ▶ The certificate policy, the CA signature algorithm to use, and the delay in producing the certificate after approving the request
- ▶ The CRL issuance period and the validity period of the CRL
- ▶ Information necessary to securely access the LDAP server to store certificates and CRLs

Open Cryptographic Services Facility (OCEF) and Open Cryptographic Enhanced Plug-ins (OCEF) provide the cryptographic facilities for PKI Services. OCEF is used to access the CA certificate and private key in RACF. OCSF is used for providing the cryptographic services.

The LDAP server is used as the public repository for issued certificates and CRLs.

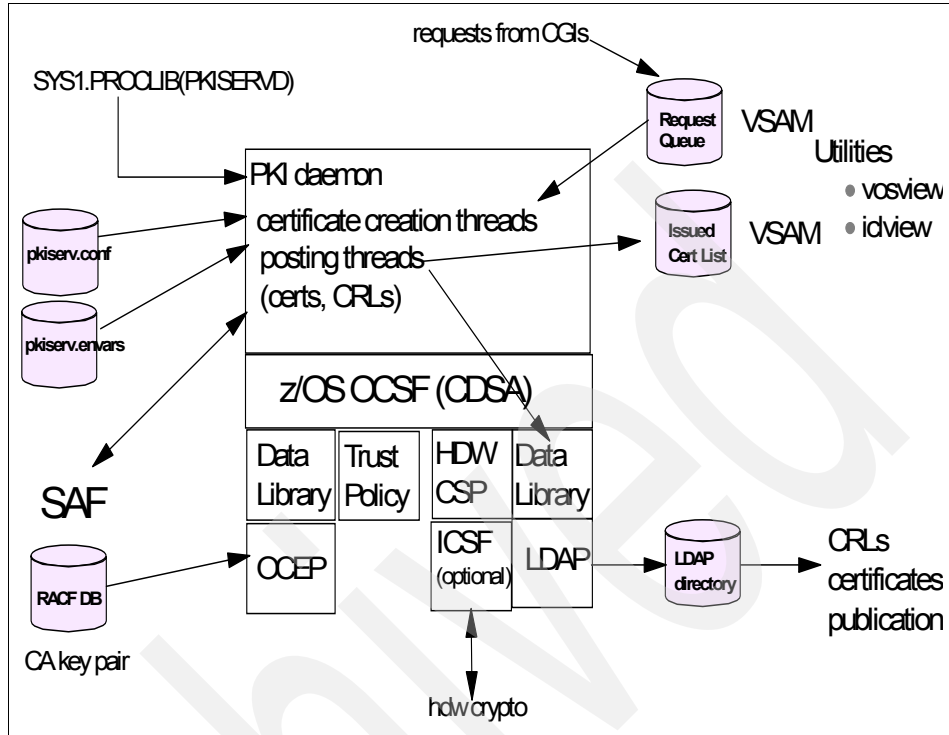


Figure 1-12 The PKI Services daemon

1.4 Security Server PKI Services enhancement in z/OS V1R4

In Release 4, the z/OS PKI Services includes support for the following:

- ▶ Support e-mail notification for completed certificate requests and expiration warnings.
- ▶ Support MAIL, STREET, and POSTALCODE distinguished name qualifiers.
- ▶ Enhance the RACDCERT command and R_PKIServ callable service to support PKCS#7 certificate chains.
- ▶ Remove clear text LDAP passwords from the pkiserv.conf file by storing them in RACF profiles.
- ▶ Use the PCI Cryptographic Coprocessor to generate key pairs, eliminating software key exposures.

- Update the list of default CERTAUTH certificates in RACF. Provide VSAM RLS for the ICL and ObjectStore VSAM data sets in support of sysplex enablement.

1.4.1 Sysplex support

In z/OS V1R3, there is no Parallel Sysplex support. Therefore, all multiple instances of PKI Services in a sysplex would be independent. For example, we needed separate databases (VSAM data sets).

In z/OS V1R4, multiple instances can share databases by accessing via VSAM record level sharing (RLS). VSAM RLS requires a Coupling Facility, couple data sets, and appropriate storage class. For more information about VSAM RLS, see *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

New sample JCL IKYMVVSAM is added to migrate existing data sets to new storage class. Specifying SharedVSAM=T in the configuration file tells PKI Services to use VSAM RLS.

Figure 1-13 on page 31 is a view of a possible Web server configuration. Each image in the sysplex has one instance of PKI Services front-ended by a Web server. One master Web server manages which image to send the work to. All images share one Coupling Facility, one set of VSAM data sets, and one LDAP directory.

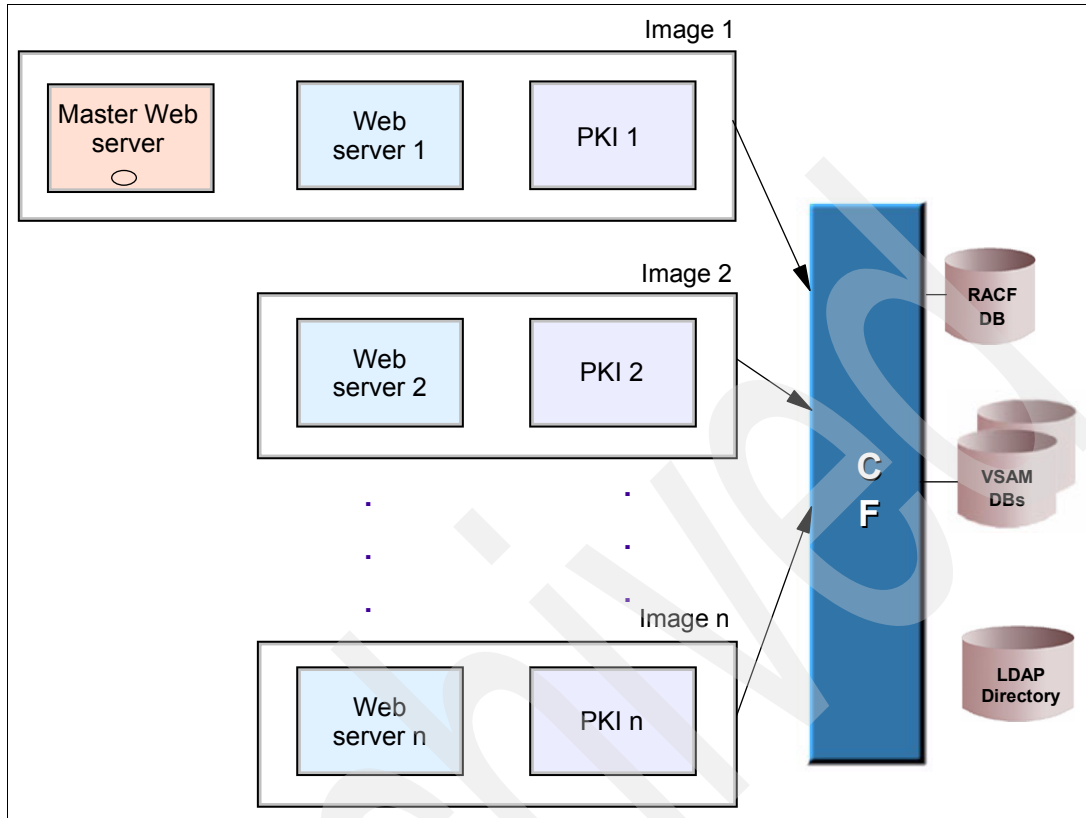


Figure 1-13 Web server configuration

To enable sysplex sharing, we move some work files into the VSAM request data set. If starting z/OS V1R4 PKI Services for the first time, those files are moved automatically. Therefore, the first image to start must have read/write access to the files listed.

HFS files moved to the VSAM request data set are following:

- ▶ Scheduler file (/var/pkiserv/pkica.j*)
- ▶ Serial file (/var/pkiserv/pkica.j*)

Old files are renamed with a .MIGRATED extension.

A new SYSTEMS ENQs is used to serialize images and a major name is SYSZPKI2. Migration and some normal processing are serialized by new ENQs.

1.4.2 Event notification via e-mail

PKI Services can now notify end users when their certificate requests are complete and when their certificates are about to expire. End users supply a notification e-mail address in the NotifyEmail field, and it is stored in the LDAP directory as the MAIL attribute, and the Communication Server's sendmail utility is used to send the message. Customer-supplied message forms are specified by path name via a configuration file, as shown in Example 1-2.

Example 1-2 User-specified message forms

```
ReadyMessageForm=/etc/pkiserv/readymsg.form
RejectMessageForm=/etc/pkiserv/rejectmsg.form
ExpiringMessageForm=/etc/pkiserv/expiringmsg.form
```

Sample message forms are shipped in the samples directory. Example 1-3 shows a sample ready message. PKI Services recognizes the following variables and substitutes values accordingly:

%%transactionid%%	The unique value returned when a certificate request has been submitted.
%%requestor%%	Name user wishes to be known by. Usually the same as the common name, except when requesting a server certificate.
%%dn%%	Subject's distinguished name.
%%notafter%%	Certificate expiration date and time.

Example 1-3 Sample ready message

```
From:dime-o-cert PKI
Subject:Certificate Ready For Pick Up
```

Attention - Please do not reply to this message as it was automatically sent by a service machine.

Dear %%requestor%%,

Thank you for choosing dime-o-cert PKI. The certificate you requested for subject %%dn%% is now ready for pickup. Please visit <http://www.dimeocert.com/PKIServ/camain.rexx> to retrieve your certificate. You will need the transaction ID listed below and your passphrase that you entered when you submitted the request.

%%transactionid%%

1.4.3 Additional distinguished name qualifier support

The subject's distinguished name (DN) is specified via qualifier name and value pairs when the request is submitted (for example, DN is CN=Jim Sweeny,O=IBM,C=US). Qualifiers are named fields in the Web pages. They can be user-supplied or hard-coded.

In z/OS V1R3, the following DN qualifiers were supported:

- ▶ CommonName
- ▶ Title
- ▶ OrgUnit
- ▶ Org
- ▶ Locality
- ▶ StateProv
- ▶ Country

In z/OS V1R4, the following qualifiers have been added:

- ▶ Email
- ▶ PostalCode
- ▶ Street

If both Email and NotifyEmail are specified, they must be equal.

1.4.4 LDAP password encryption

PKI Services posts information to LDAP using a bind DN and password.

In z/OS V1R3, bind password specified in the clear as configuration parameter AuthPWDn.

In z/OS V1R4, the server name, port, bind DN, and password may be specified in the PROXY segment of RACF profile. You can create a profile using the RDEFINE PROXY command and LDAPBIND class is used to hold these binding profiles. The profile name is specified via the BindProfilen configuration setting. Otherwise, the IRR.PROXY.DEFAULTS in the RACF FACILITY class is used.

The password is encrypted by RACF when the profile is created.

1.4.5 PKCS#7 certificate chain support

When setting up a secure server (in particular an SSL Web server), the server must be loaded with the entire certificate hierarchy, from the root CA to the end entity. (The end entity in this case is the server itself.) For short hierarchies, loading individual certificates manually is relatively simple. For longer hierarchies, this is labor-intensive and can be prone to errors.

PKCS#7 certificate packages contain the entire chain. Software that can read the entire chain eliminates the manual work. Figure 1-14 shows the certificate chains.

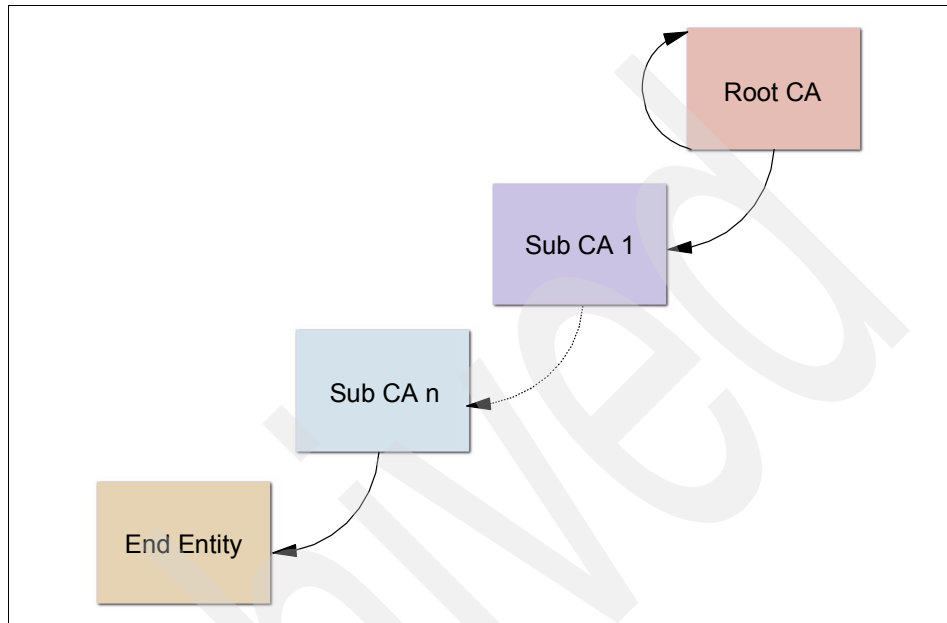


Figure 1-14 PKCS#7 certificate chains

RACF now has full support for PKCS#7 certificate packages. The RACF command to add certificates is RACDCERT. For PKCS#7 packages, the certificates are added in order, from top CA to subject (end entity).

The RACDCERT ADD command adds CA certificates to CERTAUTH if authorized by the user. It requires CONTROL authority to the FACILITY class profile IRR.DIGTCERT.ADD or RACF SPECIAL authority.

RACDCERT's *trust value* is an optional on/off switch, and certificates marked NOTRUST cannot be used. If not specified on the command, RACDCERT will determine trust of the top CA dynamically:

- | | |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| TRUST | If self-signed or signer is already trusted. |
| NOTRUST | If NOTRUST, trust of subsequent certificates inherited from signer. Inconsistencies cause NOTRUST (expired or unknown signature algorithm). |

1.4.6 Key generation via PCICC

Prior to z/OS V1R4, the RACDCERT GENCERT command generated the RSA key pair using software.

In z/OS V1R4, PCI Cryptographic Coprocessor (PCICC) is used for generation if a new PCICC keyword is specified.

- ▶ <no keyword specified>: Key generated in software, then stored in RACF DB
- ▶ ICSF specified: Key generated in software, then stored in PKDS
 - Fails in z/OS V1R4 if ICSF's PKA is not active. Does not save as a software key as it would in z/OS V1R3.
- ▶ PCICC specified: Key generated using PCICC, then stored in PKDS
 - Fails if ICSF PKA features, PCICC, or both are not active.

1.4.7 Additional default CERTAUTH

The following three new CERTAUTH certificates are added to the default list:

- ▶ VeriSign Class 1 Individual CA
- ▶ VeriSign Class 2 Individual CA
- ▶ VeriSign International Svr CA

The following two expiring CERTAUTH certificates are replaced:

- ▶ VeriSign Class 2 Primary CA
- ▶ VeriSign Class 3 Primary CA

The following defunct CERTAUTH certificate are no longer added:

- ▶ IBM World Registry™

1.4.8 Summary of z/OS PKI external characteristics as of z/OS V1R4

- ▶ This is a PKIX-compliant PKI, providing X.509 V3 certificates and producing X.509 V2 CRLs.
- ▶ The CA in the z/OS PKI can be a root CA or an intermediate CA.
- ▶ Signature algorithm is RSA. Note that this is also the algorithm supported by SSL-enabled servers and RACF in z/OS.
 - The key length for signature generation is 1024 bits.
 - The hash algorithm used for signature is MD5 or SHA-1.
- ▶ There is no OCSP support. The CRL characteristics are:
 - Access via LDAP only

- Global CRL only, no distribution point as of z/OS V1R4
- ▶ End user and administrative interaction are via HTTP/HTTPS only.
- ▶ The extensions supported in the certificates are:
 - Standard extensions
 - Authority key identifier
 - Basic constraints
 - Certificate policies
 - Key usage
 - Subject alternate name
 - Subject key identifier
 - CRL distribution points
 - Other extensions
 - The IBM hostIDmappings extension

RACF for PKI Services

This chapter assumes that readers who plan to use PKI Services already have an RACF environment established for UNIX System Services (USS).

When in this chapter we refer to *you*, we mean RACF administrators or RACF systems programmers. This chapter is intended to help people working in these roles to plug all necessary RACF controls to enable installation of PKI Services.

Almost all of the necessary RACF work to be performed to have PKI Services up and running is in the REXX exec IKYSETUP residing in SYS1.SAMPLIB and in Chapter 25 of *z/OS Security Server PKI Services Guide and Reference*, SA22-7693 and discussed in the book's chapters 4 and 25. Running this executable enables RACF people, or people assigned the task of installing PKI Services who are not familiar with the RACDCERT command, to run it, create an RACF environment, and then deal with the `httpd.conf` and `pkiserv.conf` customizations.

Although IKYSETUP is a valuable tool, we undertook a different approach in this book. First we explain how to create an RACF environment for PKI Services when undoubtedly a lot of naming standards, procedures, and RACF design style are already in place, and then run a straightforward job through which the new product seamlessly fits into the organization's RACF database.

2.1 Introduction to creating an RACF environment for new products

Good naming standards for all resources in the z/OS world are a necessary condition, but not sufficient for having neat, tidy, easy-to-manage RACF databases—this requires permanent, everyday enforcement of these standards without any compromise.

For our presentation, we assume that you have established naming standards for several types of groups and user IDs. We also assume that you already have a policy for ownership of resources.

2.1.1 RACF group structure

An important condition for a good RACF database is its group structure. It is outside the scope of this book to deal with the best way to design such a structure. However, it is sufficient to recommend a very simple structure as shown in Figure 2-1.

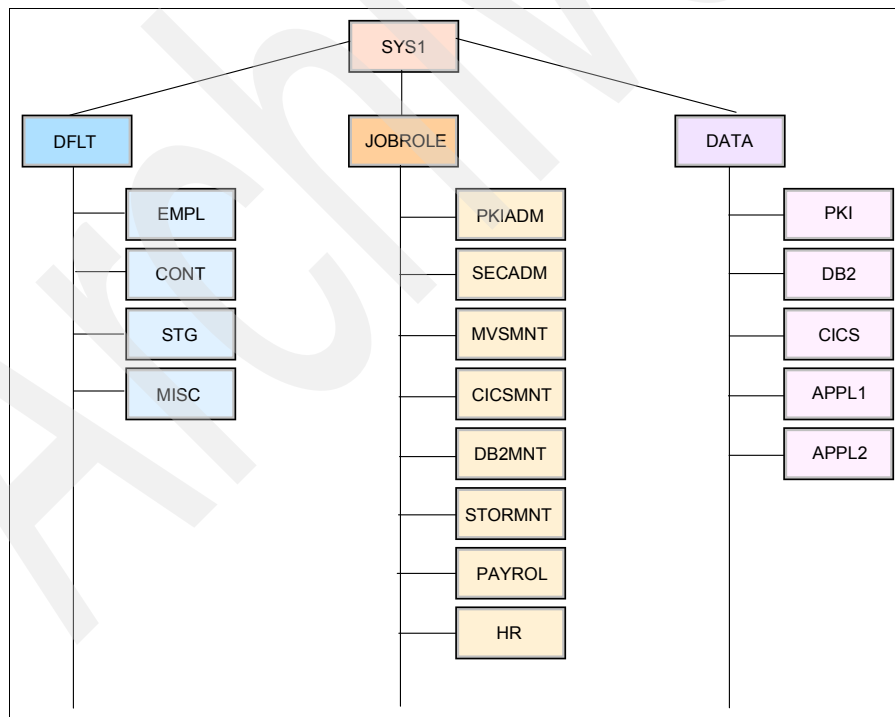


Figure 2-1 Very simple RACF group structure

Default group for started task user IDs

As part of the job role group structure implementation we create an RACF default group for all started task user IDs on our systems. You may have such a group with names such as STC, STCG, and STCGRP. Our preferred name is STG and the command to create it is:

```
AG STG SUPGROUP(DFLT) OWNER(DFLT)
```

Note: The group DFLT is a subgroup of group SYS1 and has as subgroups all default groups (for example, default group EMPL for all employees, default group CONT for all contractors and vendors, and default group STG for all started task user IDs). Default groups should never be permitted to any resource.

Job role groups

In this book we introduce the concept of *job role* RACF group. The user IDs connected to such a group should have the same set of access to various resources in order to do their everyday job. For example, the group of MVS systems programmers (for example, MVSMNT) need access ALTER to all operating system data sets and all data sets related to software products. They do not need even READ access to data sets storing data from financial, human resources, or other business applications. When an MVS systems programmer joins or leaves the team, the RACF administrator needs only to connect or remove him to or from group MVSMNT. To implement such a concept, much preliminary work is required (interviews with representatives of all areas of the enterprise, logical assessment by an RACF architect, security policy review) followed by the difficult task of actually reengineering the whole RACF group structure.

We assume that a job role group structure is implemented at least in the Information Technology Department of your organization. We will refer to the Security Administrators Group as SECADM and to the RACF Systems Programmers Group as SECMNT.

2.1.2 Machine user IDs

Started task user IDs

In this section we refer to started task user IDs (STUs) necessary to run procedures (programs for various subsystems and products). In the UNIX world the STUs are called daemons. We assume that your site has a naming standard for STUs, maybe something similar to:

- ▶ productnameTASK
- ▶ STCproductname

- ▶ productnameSTC
- ▶ productnameSTU

We prefer the last naming standard. In the framework of the job role concept for group structure, we recommend that STUs are never connected to job role groups, but placed individually on Access Lists (ALs) instead.

Miscellaneous machine user IDs

Here we refer to various machine user IDs (such as batch submission IDs, CICS default user, console IDs, surrogate IDs). They should have their own default group (such as MISC) and should be placed individually on ALs.

2.1.3 System data set profiles

Most customers have for many years used well-established naming standards for data sets. Usually all IBM system software resides in data sets prefixed with SYS1, while in-house modifications to software, as well as system data, reside in data sets prefixed with SYS2, SYSU, and so on. Our preferred prefix is SYSU.

2.1.4 Ownership

Although OWNER may not have any significance in RACF except when a decentralized administration is in place, we recommend that all resources have a meaningful OWNER (always an RACF group). For example, user IDs are owned by their default groups, system data sets profiles are owned by MVSMNT, CICS software data sets are owned by CICS systems programmers, application data sets are owned by respective business groups, security profiles (BPX, IRR in facility) are owned by SECADM or SECMT, and so on.

There is not great secret to creating a new RACF environment for a new product. Just open your Naming Standards and Procedures Cookbook and create a few user IDs and data set profiles according to the suffixes and prefixes there.

2.2 New RACF features

When installing PKI Services and its prerequisite products you have to make decisions regarding access to UNIX directories and files you will need to determine the assignment of UIDs and GIDs to various user IDs and groups. Two new features of RACF that will help you when making these decisions are:

- ▶ Access Control Lists (ACLs) introduced with z/OS 1.3
- ▶ Automatic assignment of UIDs and GIDs introduced with z/OS 1.4

2.2.1 Access control lists

Traditionally, the authorization checking for accessing to z/OS UNIX files and directories in a file system has been done using the file security packet (FSP), as shown in Figure 2-2.

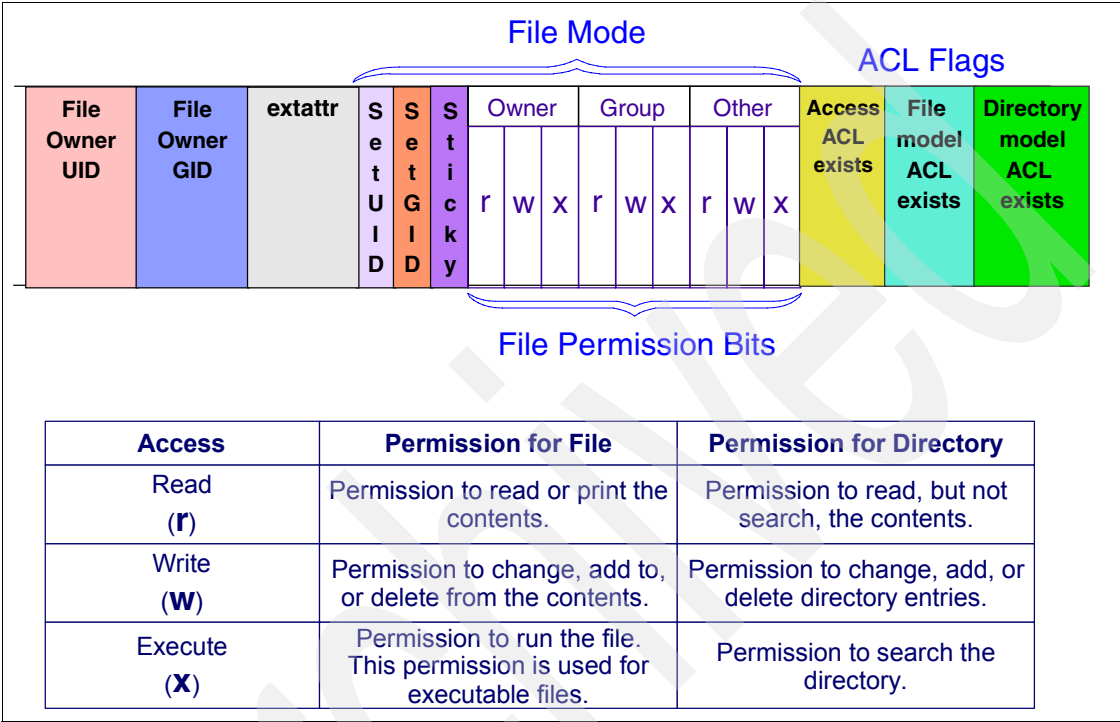


Figure 2-2 FSP with ACL flags

The FSP is stored in the file system as part of the attributes of a file or directory and is created when the file directory is created. If a security authorization is needed for a file or directory, the security packet is passed to the security product for authorization checking. The level of authorization for the file or directory through the FSP enables specification of file permission bits for file owner (user), group owner (group), and anybody else (other) but cannot permit or restrict access to other specific users and groups.

HFS and zFS files are currently protected with POSIX permission bits contained within the FSP in the file system (not in RACF).

A description of ACLs

The permission bit model does not allow granting and denying access to specific users and groups, such as is possible when using RACF profiles. This function is

now provided by the introduction of ACLs in the z/OS UNIX file system. An ACL is an SAF-owned construct that resides within the file system. ACLs are used in combination with the traditional permission bits in a modified FSP (see Figure 2-2 on page 41) to control access to z/OS UNIX files and directories by any individual users (UIDs) and groups (GIDs) in addition to the owning user and the owning group. In the framework of ACL, the traditional permission bits are called *base* ACLs, and the newly introduced ACLs are *extended* ACLs. ACLs are created and checked by RACF using two shell commands:

set fac1	Create, modify, delete ACL
get fac1	Display ACL

ACL types

In order to reduce administrative overhead, three types of extended ACLs are defined. This gives us the capability to inherit ACLs to newly created files and directories as follows:

Access	This type of ACL is used to provide protection for a file or directory.
File default	This type is a model ACL that is inherited by <i>files</i> created within the parent directory. The file default ACL is copied to a newly created file as its access ACL. It is also copied to a newly created subdirectory as its file default ACL.
Directory default	This type is a model ACL that is inherited by <i>subdirectories</i> created within the parent directory. The directory default ACL is copied to a newly created subdirectory as both its access ACL and directory default ACL.

Inheritance is the act of automatically associating an ACL with a newly created object. Administrative action is not needed.

Working with access ACLs

To create an ACL for a file, you must have one of the following security access controls:

- ▶ Be the file owner.
- ▶ Have READ access to BPX.SUPERUSER profile in class FACILITY.
- ▶ Have superuser authority (UID=0).
- ▶ Have READ access to the SUPERUSER.FILESYS.CHANGEPERMS profile in the class UNIXPRIV.

Note: The RACF UNIXPRIV class was introduced in OS/390 V1R8. It enables you to define profiles in the UNIXPRIV class to grant RACF authorization for certain z/OS UNIX privileges. By defining profiles in the UNIXPRIV class, you can grant specific superuser privileges to users who do not have superuser authority (UID=0). This enables you to minimize the number of assignments of superuser authority at your installation and reduces your security risk.

To activate the use of ACLs in z/OS UNIX file authority checks, run this RACF command to activate the new RACF class FSSEC:

```
SETROPTS CLASSACT(FSSEC)
```

You can define ACLs prior to activating the FSSEC class and display ACL information but if the FSSEC class is not active, only the traditional permission bit checks are done, even if an access ACL exists.

Note: If files are deleted, ACLs are automatically deleted.

The **getfac1** and **setfac1** commands are used to manage ACLs. A few examples follow that will help you get started. For details about these commands and other commands that support ACLs, see *z/OS UNIX System Services Command Reference, SA22-7802*.

Permit any user/group to a file (create an extended ACL)

After you have issued the TSO OMVS command and made yourself a superuser, create an ACL for the `/web/pki1/httpd.conf` file. First, list the directory with the `ls -la /web/pki1/` command. The output is shown in Example 2-1.

Example 2-1 Listing directories and files for Web server for PKI Services

```
ANTOFF:/u/antoff: >su
ANTOFF:/u/antoff: >ls -la /web/pki1/
total 768
drwxr-xr-x  8 HAIMO  SYS1      8192 Apr 16 15:37 .
drwxr-xr-x 14 HAIMO  SYS1      8192 Apr 21 13:45 ..
-rw-r--r--  1 HAIMO  OMVSGRP   635 Dec 26 15:39 expiringmsg.form
-rw-r--r--  1 HAIMO  IMWEB      9 Apr 21 13:26 httpd-pid
-rwxr-xr-x HAIMO SYS1    146400 Apr 15 11:51 httpd.conf
-rw-r--r--  1 HAIMO  SYS1      483 Apr 18 17:28 httpd.envvars
drwxr-xr-x  2 HAIMO  SYS1      8192 Apr 24 00:00 logs
drw-----  2 HAIMO  SYS1      8192 Apr 16 14:45 ocsf
-rw-r--r--  1 HAIMO  OMVSGRP 24281 Jun 10 2002 pkiexit.c
drw-----  2 HAIMO  SYS1      8192 Apr 16 15:37 pkiserv
-rw-r--r--  1 HAIMO  OMVSGRP 3597 Apr 23 13:42 pkiserv.conf
-rw-r--r--  1 HAIMO  OMVSGRP 1129 Apr 23 13:31 pkiserv.envars
```

-rw-r--r--	1	HAIMO	OMVSGRP	86357	Feb 18 12:50	pkiserv.tmp
-rw-r--r--	1	HAIMO	OMVSGRP	34175	Apr 14 09:08	pkitpsamp.c
drwxr-xr-x	3	HAIMO	SYS1	8192	Apr 9 09:51	pub
-rw-r--r--	1	HAIMO	OMVSGRP	528	Dec 26 15:39	readymsg.form
-rw-r--r--	1	HAIMO	OMVSGRP	445	Dec 26 15:39	rejectmsg.form
drwxr-xr-x	2	HAIMO	SYS1	8192	Apr 24 00:00	reports
drw-----	2	HAIMO	SYS1	8192	Apr 14 15:11	sec

We grant read/write permissions to user ID PKISTU and group PKIADM using the **setfac1** command:

```
setfac1 -m user:PKISTU:rw-,group:PKIADM:rw- /web/pki1/httpd.conf
```

The -m option modifies ACL entries or adds them if they do not exist.

Example 2-2 shows how we displayed this file with the **getfac1** command.

Example 2-2 Output from getfac1 command for the httpd.conf file

```
ANTOFF:/u/antoff: >getfac1 /web/pki1/httpd.conf
#file: /web/pki1/httpd.conf
#owner: HAIMO
#group: SYS1
user::rwx
group::r-x
other::r-x
user:PKISTU:rw-
group:PKIADM:rw-
```

Note: User ID HAIMO has UID(0). If (by mistake) you try to permit access to any file owned by HAIMO to another superuser having UID(0) (for example, WEBSTU), HAIMO and not WEBSTU will be displayed (Example 2-3, row 8).

Example 2-3 shows again displaying the directory /web/pki1/.

Example 2-3 Output from ls -la showing the plus sign for the httpd.conf file

```
ANTOFF:/u/antoff: >ls -la /web/pki1/
total 768
drwxr-xr-x  8 HAIMO  SYS1      8192 Apr 16 15:37 .
drwxr-xr-x 14 HAIMO  SYS1      8192 Apr 21 13:45 ..
-rw-r--r--  1 HAIMO  OMVSGRP   635 Dec 26 15:39 expiringmsg.form
-rw-r--r--  1 HAIMO  IMWEB      9 Apr 21 13:26 httpd-pid
-rwxr-xr-x+  1 HAIMO  SYS1    146400 Apr 15 11:51 httpd.conf
-rw-r--r--  1 HAIMO  SYS1      483 Apr 18 17:28 httpd.envvars
drwxr-xr-x  2 HAIMO  SYS1      8192 Apr 24 00:00 logs
drw-----  2 HAIMO  SYS1      8192 Apr 16 14:45 ocsf
-rw-r--r--  1 HAIMO  OMVSGRP 24281 Jun 10 2002 pkiexit.c
```

drw-----	2	HAIMO	SYS1	8192	Apr 16 15:37	pkiserv
-rw-r--r--	1	HAIMO	OMVSGRP	3597	Apr 23 13:42	pkiserv.conf
-rw-r--r--	1	HAIMO	OMVSGRP	1129	Apr 23 13:31	pkiserv.envars
-rw-r--r--	1	HAIMO	OMVSGRP	86357	Feb 18 12:50	pkiserv.tmp1
-rw-r--r--	1	HAIMO	OMVSGRP	34175	Apr 14 09:08	pkitpsamp.c
drwxr-xr-x	3	HAIMO	SYS1	8192	Apr 9 09:51	pub
-rw-r--r--	1	HAIMO	OMVSGRP	528	Dec 26 15:39	readymsg.form
-rw-r--r--	1	HAIMO	OMVSGRP	445	Dec 26 15:39	rejectmsg.form
drwxr-xr-x	2	HAIMO	SYS1	8192	Apr 24 00:00	reports
drw-----	2	HAIMO	SYS1	8192	Apr 14 15:11	sec

The plus sign in the first column of the httpd.conf file indicates that an ACL entry exists.

Delete a user or group from an extended ACL

Option -x removes a user or group from ACL, as shown in Example 2-4 in which we removed user PKISTU:

```
setfacl -x user:PKISTU /web/pki1/httpd.conf
```

Example 2-4 Removing user ID PKISTU from ACL for in the httpd.conf file

```
getfacl /web/pki1/httpd.conf
#file: /web/pki1/httpd.conf
#owner: HAIMO
#group: SYS1
user::rwx
group::r-x
other::r-x
group:PKIADM:rw-
```

Change base and extended ACL

The -s option replaces the contents of an ACL with the entries specified on the command line. It requires that the base permissions be specified. The base permissions (the traditional UNIX bits) are specified similarly to extended ACL entries, except that there is no user or group name qualifier.

We want to disallow access to everybody except the file owner, who will have read-only access:

```
setfacl -s user::r--,group:---,other:--- /web/pki1/httpd.conf
```

Example 2-5 shows the results.

Example 2-5 Setting base permissions

```
getfacl /web/pki1/httpd.conf
#file: /web/pki1/httpd.conf
```

```
#owner: HAIMO
#group: SYS1
user::r--
group:---
other:---
```

The same option can be used to simultaneously change the base ACL and add extended ACL:

```
setfacl -s user::rwx,group::r-x,other:--- , user:PKISTU:r--
/web/pki1/httpd.conf
```

Example 2-6 shows the results.

Example 2-6 Simultaneously setting base and extended permissions

```
getfacl /web/pki1/httpd.conf
#file: /web/pki1/httpd.conf
#owner: HAIMO
#group: SYS1
user::rwx
group::r-x
other:---
user:PKISTU:r--
```

Delete an access ACL

The `-D a` option specifies that the access ACL is to be deleted. The base ACL remains unchanged. When a file is deleted, its ACL is deleted automatically; there is no extra administrative effort required.

```
setfacl -D a /web/pki1/httpd.conf
```

Example 2-7 shows the results.

Example 2-7 Deleting the whole access ACL

```
getfacl /web/pki1/httpd.conf
#file: /web/pki1/httpd.conf
#owner: HAIMO
#group: SYS1
user::r--
group:---
other:---
```

The options for the **setfacl** command work exactly the same way for directories:

```
setfacl -m user:PKISTU:r-- /web/pki1/
```

Example 2-8 on page 47 shows the results.

Example 2-8 Allow user ID PKISTU to have r-- in the /web/pki1/ directory

```
getfacl /web/pki1/  
#file: /web/pki1/  
#owner: HAIMO  
#group: SYS1  
user::rwx  
group::r-x  
other::r-x  
user:PKISTU:r--
```

As we see, the creation and deletion of access ACLs resembles the creation and deletion of RACF discrete data set profiles.

The file default ACL and the directory default ACL may be used to facilitate and streamline the security administrator's effort by providing models for access for newly created files or directories.

Working with default ACLs

Create file default ACL and directory default ACL

The **setfacl** command uses a second keyword to manage file and directory default ACLs:

d	for directory
f	for file

To create a directory default ACL for directory /u/antoff/ that allows READ (read content of directory) and EXECUTE (search directory) access to group PKIADM every time a subdirectory of /u/antoff/ is created, use the command:

```
setfacl -s u::rwx,g::---,o::---,d:g:PKIADM:r-x /u/antoff/
```

To display the result, **getfacl** must be used with the -d keyword as shown in Example 2-9.

Example 2-9 Creating a directory default ACL for directory /u/antoff/

```
getfacl -d /u/antoff/  
#file: /u/antoff/  
#owner: ANTOFF  
#group: SYS1  
default:group:PKIADM:r-x
```

Attention: If we now try to create a file default ACL for the /u/antoff/ directory that allows READ access to user ID TRAUNER every time a new file in /u/antoff/ is created with **setfacl -s u::rwx,g::---,o::---,f:u:trauner:r-- /u/antoff/** then our newly created *directory* default ACL will be deleted.

The -m option is more flexible when we do not want to change the basic ACL. To test this, we deleted the most recently created f default ACL with:

```
setfacl -D f /u/antoff/
```

and issued in succession:

```
setfacl -m d:g:PKIADM:r-x /u/antoff/  
setfacl -m f:u:trauner:r-- /u/antoff/
```

This time **getfacl** is used with the -df keywords, and the result is shown in Example 2-10.

Example 2-10 Creating directory and file default ACLs for the /u/antoff/ directory

```
getfacl -df /u/antoff/  
#file: /u/antoff/  
#owner: ANT OFF  
#group: SYS1  
user::rwx  
group:---  
other:---  
fdefault:user:TRAUNER:r--  
default:group:PKIADM:r-x
```

The following command removes both f and d defaults ACLs:

```
setfacl -D df /u/antoff
```

For details about the ISHELL interface for ACLs, refer to sections 8.1.10 through 8.1.12 of *z/OS Version 1 Release 3 and 4 Implementation*, SG24-6581.

File access authorization checking

RACF uses the permission bits, the access ACL, and the following UNIXPRIV class profiles to determine whether the user is authorized to access the file with the requested access level:

- ▶ SUPERUSER.FILESYS
- ▶ RESTRICTED.FILESYS.ACCESS
- ▶ SUPERUSER.FILESYS.ACLOVERRIDE

After RACF has checked the authorization, it returns control to the file system.

SUPERUSER.FILESYS profile

The SUPERUSER.FILESYS profile in the UNIXPRIV class has three access levels that allow access to z/OS UNIX files as follows:

READ	Allows a user to read any local file and to read or search any local directory.
-------------	---------------------------------------------------------------------------------

UPDATE	Allows a user to write to any local file and includes privileges of READ access.
CONTROL/ALTER	Allows a user to write to any local directory and includes privileges of UPDATE access.

RESTRICTED.FILESYS.ACCESS profile

The RESTRICTED attribute (*z/OS Security Server RACF Security Administrator's Guide*, SA22-7683) prevents a user from gaining access to a protected resource, other than a z/OS UNIX file system resource, unless the user is specifically authorized on the access list. Global access checking, the ID(*) entry on the access list, and the UACC will not be honored to allow a restricted user to access a protected resource.

The existence of profile RESTRICTED.FILESYS.ACCESS prevents a user having the RESTRICTED attribute from gaining access to z/OS UNIX files by virtue of permission bits for *other*. Place restricted users or their groups on the access list of this profile with READ access to enable them to access files through the permission bits for “other”:

```
PE RESTRICTED.FILESYS.ACCESS CL(UNIXPRIV) ID(restricted users) ACC(R)
```

This does not grant the user access to any files. It just enables the “other” bits to be used in access decisions for this user.

SUPERUSER.FILESYS.ACLOVERRIDE profile

Any user who is not a superuser with UID(0) or the file owner and is denied access through the ACL, can still access a file system resource if the user has sufficient authority to the SUPERUSER.FILESYS resource in the UNIXPRIV class. Therefore, to prevent this, you can force RACF to use your ACL authorizations to override a user's SUPERUSER.FILESYS authority by defining the following profiles:

```
RDEF UNIXPRIV SUPERUSER.FILESYS.ACLOVERRIDE UACC(NONE)
```

Note: This describes the relationship between the existing SUPERUSER.FILESYS profile and the new SUPERUSER.FILESYS.ACLOVERRIDE profile. Either profile could get checked for a file; it depends upon the presence of an ACL for the file and the contents of the ACL for granting access.

For exception cases, permit the user or group to SUPERUSER.FILESYS.ACLOVERRIDE with whatever access level would have been required for SUPERUSER.FILESYS as follows:

```
PERMIT SUPERUSER.FILESYS.ACLOVERRIDE CLASS(UNIXPRIV) ID(decentralized  
admin)+ ACCESS(READ)
```

SUPERUSER.FILESYS.CHANGEPERMS profile

As an enhancement to superuser granularity, when a user who is not the owner or a superuser but has READ access to the SUPERUSER.FILESYS.CHANGEPERMS profile in the UNIXPRIV class, he is authorized to change the file mode in the same manner as a user having UID(0) and to use the **setfac1** command to manage access ACLs for any file.

For more information about the new RESTRICTED.FILESYS.ACCESS, SUPERUSER.FILESYS.ACLOVERRIDE, and SUPERUSER.FILESYS.CHANGEPERMS UNIXPRIV profiles, refer to *z/OS V1R3 Security Server RACF Security Administrator's Guide*, SA22-7683.

2.2.2 Automatic assignment of UID/GID

Enhancements in z/OS V1R4 have been made in the way that UIDs and GIDs can be assigned by RACF. They can be assigned automatically to new users and prevented from sharing or allowing to be shared.

New keywords have been added to the SEARCH command in order to determine the set of users assigned to a given UID or the groups assigned to a given GID.

The RACF UNIXMAP class is used to enable the system to quickly look up a user ID from a UID, or a group name from a GID. An enhanced RACF database organization can now perform this conversion quickly without requiring the mapping profiles in the UNIXMAP class. If desired, you should use the IRRIRA00 utility to convert the RACF database to the new organization.

RACF database and application identity mapping

To use the new RACF profile, SHARED.IDS, and the new SEARCH keywords, the RACF database must have application identity mapping (AIM) stage 2 or 3 implemented. You can convert your RACF database to application identity mapping stage 3 using the IRRIRA00 conversion utility. See the *z/OS Security Server RACF System Programmer's Guide*, SA22-7681, for information about running the IRRIRA00 conversion utility.

If your installation is new to RACF and you are not running any releases prior to OS/390 Version 2 Release 10, you will automatically take advantage of application identity mapping at the stage 3 level without running the IRRIRA00 conversion utility, and you will not have to use VLF and UNIXMAP to achieve improved performance.

IRRIRA00 utility

This utility was new in OS/390 V2R10. It converts an existing RACF database to application identity mapping functionality using a four-staged approach.

With a database created at OS/390 Release 10 or higher, RACF's application identity mapping uses an alias index to associate user and group names with:

- ▶ Lotus® Notes® for z/OS
- ▶ Novell Directory Services for OS/390
- ▶ z/OS UNIX identities

RACF typically uses mapping profiles for this purpose with databases created before Release 10. However, you can use the IRRIRA00 utility to convert the database to work with an alias index instead. The conversion consists of a series of stage transitions from zero to three. RACF uses the database mapping information differently for each stage.

AIM stage 3

In stage 3, RACF locates application identities, such as UIDs and GIDs, for users and groups by using an alias index that is automatically maintained by RACF. This enables RACF to more efficiently handle authentication and authorization requests from applications such as z/OS UNIX than was possible using other methods, such as the UNIXMAP class and VLF. When your installation reaches stage 3 of application identity mapping, you will no longer have UNIXMAP class profiles on your system, and you can deactivate the UNIXMAP class and remove VLF classes IRRUMAP and IRRGMAP.

Search enhancements to map UIDs and GIDs

The use of the new SEARCH keywords, UID and GID, requires having at least AIM Stage 2 installed. However, it does not require any particular authority. When a UID or GID is specified, all other keywords except CLASS are ignored. The RACF search command is:

```
SEARCH CLASS(USER) UID(0)
```

Using the UID keyword, as shown in the Example 2-11, you can obtain a list of all users assigned to the specified UID.

Example 2-11 Display all users with a UID=1

```
SEARCH CLASS(USER) UID(1)
LDAPSRV
MSYSLDAP
```

Using the GID keyword, as shown in Example 2-12 on page 52, you can obtain a list of all groups assigned to the specified GID.

Example 2-12 Display all groups with a GID=1

```
SEARCH CLASS(GROUP) GID(1)
OMVSGRP
```

Shared UID prevention

To prevent several users from having the same UID number, a new RACF SHARED.IDS profile was introduced in the UNIXPRIV class. This profile acts as a system-wide switch to prevent assignment of a UID that is already in use.

To enable shared UID prevention, it is necessary to define the new SHARED.IDS profile in the UNIXPRIV class, as shown in the following command:

```
RDEF UNIXPRIV SHARED.IDS UACC(NONE)
```

After the SHARED.IDS profile has been defined and the UNIXPRIV class refreshed, it will not allow a UID already in use to be assigned. The same is true for GIDs: It does not allow a GID to be shared between different groups.

Shared UID examples

In the following example, we created a user USER1 with UID(7). Then, we tried to define a new user USER2 with the same UID(7). The following error message was issued:

```
IRR52174I Incorrect UID 7. This value is already in use by USER1.
```

The following error message appears if you try to specify more than one user in an ADDUSER command request:

```
IRR52185I The same UID cannot be assigned to more than one user.
```

Existing shared UIDs

The use of this new functionality does not affect pre-existing shared UIDs. They remain as shared when you install the new support. If you want to eliminate sharing of the same UID, you must clean them up separately. The release provides two new IRRICE reports to find the existing shared UIDs and GIDs. See member IRRICE in SYS1.SAMPLIB and search for UID and GID.

Even if the SHARED.IDS profile is defined, you may still require some UIDs to be shared and others not to be shared. For example, you may require multiple superusers with a UID(0). It is possible to do this using the new SHARED keyword in the OMVS segment of the ADDUSER, ALTUSER, ADDGROUP, and ALTGROUP commands.

To allow an administrator to assign a non-unique UID or GID using the SHARED keyword, you must grant that administrator at least READ access to SHARED.IDS profile, as follows:

```
PE SHARED.IDS CLASS(UNIXPRIV) ID(SECADM) ACCESS(READ)
```

Now administrators will be able to assign the same UID or GID to multiple users, using the SHARED KEYWORD, as follows:

```
ADDUSER (USERA USERB) OMVS(UID(7) SHARED)
```

Automatic UID/GID assignment

UIDs and GIDs can be assigned automatically by RACF to new users, making it easier to manage the process of assigning UIDs and GIDs to users and groups. Previously, this was a manual process and did not guarantee the uniqueness of the UID and GID for every user.

Using a new AUTOUID keyword with the ADDUSER and ALTUSER commands, an unused UID is assigned to the new or modified user. Using the AUTOGID keyword on ADDGROUP and ALTGROUP commands, a GID is assigned automatically to the new or modified group.

The use of automatic UID/GID requires:

- ▶ AIM stage 2 or 3; otherwise, an IRR52182I message is issued and the automatic assignment attempt fails:

```
IRR52182I Automatic UID assignment requires application identity mapping to be implemented
```
- ▶ A SHARED.IDS profile must be defined; otherwise, an IRR52183I message is issued and the attempt fails:

```
IRR52183I Use of automatic UID assignment requires SHARED.IDS to be implemented
```
- ▶ The profile BPX.NEXT.USER in class FACILITY must be defined. Otherwise, an IRR52179I message is issued and attempts fail:

```
IRR52179I The BPX.NEXT.USER profile must be defined before you can use automatic UID assignment.
```

The definition of the BPX.NEXT.USER FACILITY class profile has the following syntax:

```
RDEF FACILITY BPX.NEXT.USER APPLDATA('UID number/GID number'),
```

In this syntax, APPLDATA consists of two qualifiers separated by a forward slash and surrounded by small quotes ('/'). The qualifier to the left of the slash character specifies starting UID number or range of UID numbers. The qualifier to the right of the slash specifies the starting GID number or range of

GID numbers. Qualifiers can be null or specified as NOAUTO to prevent automatic assignment of UIDs or GIDs.

The starting value is the value RACF attempts to use in ID assignment after determining that the ID is not in use. If it is in use, the value is incremented until an appropriate value is found.

The maximum value valid in the APPLDATA specification is 2,147,483,647. If this value is reached or a candidate UID/GID value has been exhausted for the specified range, subsequent automatic ID assignment attempts fail and message IRR52181I is issued.

Note: Be careful because APPLDATA is verified at time of use, not at definition time. If a syntax error is encountered at the use of the auto assignment, an IRR52187I message is issued and the attempt fails.

Automatic assignment example

In Example 2-13, we defined the APPLDATA for a range of values from 5 to 70000 for UIDs and from 3 to 3000 for GIDs. USERA and USERB are added using the automatic assignment of UID. The range of automatic UID assignment starts with 5, so USERA is assigned to UID(5), which was free. UID(6) and UID(7) were already assigned before we started our example, so the next free UID is 8. USERB is assigned to UID(8), as shown in Example 2-13.

Example 2-13 Defining the APPLDATA

```
RDEFINE FACILITY BPX.NEXT.USER APPLDATA('5-70000/3-30000')
```

```
ADDUSER USERA OMVS(AUTOUID)
```

```
IRR52177I User USERA was assigned an OMVS UID value of 5.
```

```
ADDUSER USERB OMVS(AUTOUID)
```

```
IRR52177I User USERB was assigned an OMVS UID value of 8.
```

RACF extracts the APPLDATA from the BPX.NEXT.USER and parses out the starting value. It checks whether it is already in use and if so, the value is incremented and checked again until an unused value is found. When a free value is found, it assigns the value to the user or group and replaces the APPLDATA with the new starting value, which is the next potential value or the end of the range.

In our example, that means that if UID(6) becomes free after UID(7) is assigned to USERB, RACF will start checking from UID(8) in the next assignment, so it will not assign UID(6). But you can change the APPLDATA and modify the starting value. The APPLDATA can be changed using this command:

```
RALTER FACILITY BPX.NEXT.USER APPLDATA('2000/500')
```

Note: Automatic assignment of UIDs and GIDs fails if you specify a list of users to be defined with the same name or if you specify the SHARED keyword. Also, AUTOUID or AUTOGID is ignored if UID or GID is also specified.

APPLDATA examples

Here are some examples of correct and incorrect APPLDATA specifications:

- ▶ **Good data**
 - 1/0
 - 1-50000/1-20000
 - NOAUTO/100000
 - /100000
 - 10000-20000/NOAUTO
 - 10000-20000/
- ▶ **Bad data**
 - 123B
 - /
 - 2147483648 /* higher than max UID value */
 - 555/1000-900

If you have an incorrect specification and attempt to use AUTOUID on an ADDUSER command, the following message is issued:

```
IRR52187I Incorrect APPLDATA syntax for the BPX.NEXT.USER profile.
```

For more information about automatic assignment, see the *SecureWay Security Server RACF Security Administrator's Guide*, SA22-7683.

2.3 Setting up RACF environment for PKI prerequisites

The implementation of PKI Services requires that z/OS UNIX System Services (USS) is active. Therefore, we expect that most of the USS RACF controls are in place as per the content of member BPXISEC1 in SYS1.SAMPLIB.

The prerequisite products for PKI Services are z/OS HTTP Server or Web server, Lightweight Directory Access Protocol (LDAP) server, Open Cryptographic Services Facility (OCSF), and Open Cryptographic Enhanced Plug-ins (OCEP).

Integrated Cryptographic Services Facility (ICSF) and z/OS Communications Server's sendmail utility are optional.

2.3.1 z/OS UNIX level security

We recommend you have z/OS level of UNIX security on your system. More details about the two possible levels of security (UNIX level of security and z/OS level of security) can be found in *z/OS UNIX System Services Planning*, GA22-7800.

The RACF controls necessary to establish the z/OS UNIX level of security are:

- ▶ Program control
- ▶ Daemon and server control

Program control

For program control you have to create profile ** in class PROGRAM and then ADDMEM the following libraries:

```
RDEF PROGRAM ** OWNER(MVSMNT) UACC(READ)
RALT PROGRAM ** ADDMEM('CEE.SCEERUN'//NOPADCHK +
'CBC.SCLBDLL'//NOPADCHK +
'GLD.SGLDLNK'//NOPADCHK +
'GSK.SGSKLOAD'//NOPADCHK +
'SYS1.CSSLIB'//NOPADCHK +
'TCPIP.SEZALOAD'//NOPADCHK +
'SYS1.LINKLIB'//NOPADCHK +
'CSF.SCSFMODE'//NOPADCHK +
'CSF.SCSFMOD1'//NOPADCHK)
```

Note: You may wish to convert from an existing profile * to profile ** to prevent the output of RLIST PROGRAM * ALL from displaying all profiles created in class PROGRAM and displaying only the content of profile ** instead. But be careful: You have to ADDMEM to profile ** all already-ADDMEMed libraries to profile * (if additional to the ones above) before issuing RDEL PROGRAM * and SETR WHEN(PROGRAM) REFRESH.

You also might ADDMEM the following libraries to profile ** in PROGRAM, replacing the italics (*hlq*) with HLQs that are used at your site, most likely SYS1:

- ▶ If using the Resource Measurement Facility (RMF), *hlq.SERBLINK*
- ▶ If using DB2, *hlq.SDSNLOAD* and *hlq.SDSNEXIT*
- ▶ If using MQSeries, *hlq.SESQLINK*
- ▶ If using Run-Time Library Services (RTLS), *hlq.SCEERTLS*
- ▶ If you are obtaining an IEATDUMP by setting the SysDumpName directive and setting the Recovery directive to Msg/Dump, Normal, or Full, *hlq.MIGLIB*

If you failed to make a load library program controlled, the following message appears in the SDSF LOG. (Example 2-14 is for a DB2 load library.)

Example 2-14 Message if load library program not controlled

```
ICH420I PROGRAM DSNAOCLI FROM LIBRARY DB2H7.SDSNLOAD CAUSED THE
ENVIRONMENT TO BECOME UNCONTROLLED.
BPXP014I ENVIRONMENT MUST BE CONTROLLED FOR DAEMON (BPX.DAEMON)
PROCESSING.
BPXM023I (ITSOLDP) 338
GLD5002A The __passwd() function failed; not loaded from a program
controlled library.
```

To rectify, just ADDMEM the library to profile ** in class PROGRAM.

Daemon and server control

The command to create a profile for restricting your daemons from being able to change their identity is:

```
RDEF FACILITY BPX.DAEMON UACC(NONE) OWNER(SECADM)
```

Place on the access list only daemons that are allowed to change their identity:

```
PE BPX.DAEMON CL(FACILITY) ID(daemon1 daemon2 etc) ACC(READ)
```

Note: Replace *daemon1*, *daemon2*, *etc* with RACF user IDs for your respective daemons.

The command to create a profile to set the scope of z/OS resources that the server can access when acting as a surrogate for its clients is:

```
RDEF FACILITY BPX.SERVER UACC(NONE) OWNER(SECADM)
```

Then place on the access list only daemons that may act as surrogates:

```
PE BPX.SERVER CL(FACILITY) ID(daemon1 etc) ACC(READ)
PE BPX.SERVER CL(FACILITY) ID(daemon2 etc) ACC(UPDATE)
```

READ

Both the server ID and the RACF ID of the client must be authorized to access resources and the client must supply a password.

UPDATE

The server acts as a surrogate of the client (uses the identity and access granted to the client without the client's password being specified).

2.3.2 RACF for Web server

Note: In this book we use the term Web server instead of z/OS HTTP server.

Started task user ID

The command to create a started task user ID for your Web server is:

```
ADDUSER WEBSTU DFLTGRP(STG) OMVS(UID(0) HOME('/usr/lpp/internet')
PROGRAM('/bin/sh'))
```

Note: To define your Web server user ID with a non-zero UID and according to your naming standard for STUs, give this user ID appropriate permissions to directories and files:

```
SETFACL -m u:WEBSTU:r-x /usr/lpp/internet/sbin/httpd_V5R3M0
SETFACL -m u:WEBSTU:r-x /web/pki1/httpd.conf
SETFACL -m u:WEBSTU:rw- /web/pki1/logs
SETFACL -m d:u:WEBSTU:rw- /web/pki1/logs
SETFACL -m f:u:WEBSTU:rw- /web/pki1/logs
SETFACL -m u:WEBSTU:rw- /web/pki1/httpd-pid
```

The Web server's non-zero user ID must be given read/execute access to the security DLLs, read/write access to the key database file, and read access to the stash file.

Profile in class STARTED

To create a profile in class STARTED for your Web server procedure, use:

```
RDEF STARTED WEB*.** STDATA(USER(WEBSTU) GROUP(STG))
```

Daemon and server control

The commands to control your Web server started task user ID are:

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(WEBSTU) ACCESS(READ)
PERMIT BPX.SERVER CLASS(FACILITY) ID(WEBSTU) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Access to profiles in class CSFSERV

Use this access only if ICSF is active. (See 2.3.5, "RACF for ICSF" on page 60.)

```
PE CSF* CL(CSFSERV) ID(WEBSTU) ACC(R)
```


2.3.3 RACF for OCSF and OCEP

The use of Open Cryptographic Services Facility (OCSF) and Open Cryptographic Enhanced Plug-ins (OCEP) is controlled by fixed-name profiles in class FACILITY:

CDS.CSSM	Authorizes access to OCSF
CDS.CSSM.CRYPTO	Authorizes access to Cryptographic Service Provider
CDS.CSSM.DATALIB	Authorizes access to Data Library (DL) Service Provider

All of these resources probably have the same access list, so only one profile may be defined:

```
RDEF FACILITY CDS.** UACC(NONE) OWNER(SECADM)
```

2.3.4 RACF for LDAP

The RACF setup for the LDAP server consists of the following commands:

1. Started task user ID

```
ADDUSER LDAPSTU DFLTGRP(STG) NAME(LDAP Started Task User') +  
OMVS(AUTOUID) HOME('/') PROG('/bin/sh'))
```

2. Profile in class STARTED

```
RDEF STARTED LDAP*.** STDATA(USER(LDAPSTU) GROUP(STG))  
SETR RACLIST(STARTED) REFR
```

3. Access to DB2 from TSO and BATCH

```
PE DB2subsystem.BATCH ID(LDAPSTU) ACC(R)
```

4. Daemon and server control

```
PE BPX.DAEMON CL(FACILITY) ID(LDAPSTU) ACC(R)  
PE BPX.SERVER CL(FACILITY) ID(LDAPSTU) ACC(U)  
SETR RACLIST(FACILITY) REFR
```

5. Access to OCSF

```
PE CDS.** CL(FACILITY) ID(LDAPSTU) ACC(R)
```

6. Access to profiles in class CSFSERV

```
PE CSF* CL(CSFSERV) ID(LDAPSTU) ACC(R)
```

This access is needed only if ICSF is active. Refer to 2.3.5, “RACF for ICSF” on page 60 for more information about class CSFSERV.

2.3.5 RACF for ICSF

Using ICSF is recommended but not a prerequisite for PKI Services. You can install and configure ICSF before or after setting up PKI Services using these commands:

1. Started task user ID

```
ADDUSER ICSFSTU DFLTGRP(STG) NOPASSWORD NAME('ICSF Started task user')
```

2. ICSF data sets

These data sets contain Cryptographic Keys (CKDS) and Public Keys (PKDS): ICSF.CKDS and ICSF.PKDS.

```
AG ICSF SUPGROUP(DATA) OWNER(DATA)
AD 'ICSF.** OWNER(SECADM) UACC(NONE)
```

3. Profile in class STARTED

```
RDEF STARTED ICSF.** STDATA(USER(ICSFSTU) GROUP(STG))
```

4. General Resource Class CSFKEYS

Profiles in this class protect labels for keys in the format:

```
RDEF CSFKEYS label UACC(NONE) AUDIT(ALL)
PE label CL(CSFKEYS) ID(SECADM) ACC(R)
```

Note: You may have to create one or more job role groups in addition to SECADM for key management. Also, you may decide to create a common generic profile for all key labels:

```
RDEF CSFKEYS ** UACC(NONE) OWNER(SECADM) AUDIT(ALL)
```

They permit one or several job role groups access.

5. General Resource Class CSFSERV

Profiles in this class protect cryptographic services in fixed-name format:

```
RDEF CSFSERV service-name UACC(NONE) OWNER(SECADM) AUDIT(ALL)
PE service-name CL(CSFSERV) ID(SECADM) ACC(R)
```

Similar to CSFKEYS, you may define a common generic profile:

```
RDEF CSFSERV ** UACC(NONE) OWNER(SECADM) AUDIT(ALL) or
RDEF CSFSERV CSF* UACC(NONE) OWNER(SECADM) AUDIT(ALL)
```

For all profile names in CSFSERV class, refer to Chapter 3 of the *ICSF Administrator's Guide*, SA22-7521.

If you have not activated CSFKEYS and CSFSERV, issue the commands:

```
SETR CLASSACT(CSFKEYS CSFSERV)
SETR RACLIST(CSFKEYS CSFSERV)
SETR RACLIST(CSFKEYS CSFSERV) REFR
```

2.4 Setting up the RACF environment for PKI Services

In this section we discuss how to set up the RACF environment for PKI Services.

2.4.1 Add RACF groups for PKI Services

PKI administrator group

PKI Services administrators play a very powerful role in your organization. The decisions they make when managing certificates and certificate requests determine who will access your computer systems and what privileges they will have when doing so. We recommend appointing as PKI administrators one or more of your RACF administrators after some training, mostly on the use of the RACDCERT command.

All RACF user IDs and groups used in PKI Services must have OMVS segments. The command to add a specific job role group for PKI administrators is:

```
AG PKIADM SUPGROUP(JOBROLE) OMVS(AUTOGID)
CO (userid1 userid2 etc) GROUP(PKIADM)
```

If you need to assign an OMVS segment to your existing RACF admin group, issue the following command:

```
ALG SECADM GID(AUTOGID)
```

PKI started task user ID group

If you need to assign an OMVS segment to your existing RACF default group for started task user IDs, issue the following command:

```
ALG STG GID(AUTOGID)
```

2.4.2 Adding RACF user IDs for PKI Services

PKI started task user ID

The command to create the user ID who runs the PKI procedure (see the PKI procedure in SYS1.PROCLIB) is:

```
AU PKISTU NAME('PKI SRVS DAEMON') DFLTGRP(STG) OWNER(STG) NOPASSWORD +  
OMVS(AUTOUID HOME('/web/pki1/')) PROG('/bin/sh') ASSIZE(256000000)  
THREADS(512))
```

The ASSIZE and THREADS values are recommended, but you have to increase them if they are not enough for your workload. The full list of keywords specifying various limits in the OMVS segment is:

ASSIZEMAX	Maximum address space size
CPUTIMEMAX	Maximum CPU time
FILEPROCMAX	Maximum number of files per process
MMAPAREAMAX	Maximum memory map size
PROCUSERMAX	Maximum number of processes per UID
THREADSMAX	Maximum number of threads per process

The default value and their ranges are shown in Table 2-1.

Table 2-1 Parameters and their values in OMVS segment

Parameter	Default value	Range	Unit
ASSIZEMAX	209,715,200	10M-2G	byte
CPUTIMEMAX	1000	7-2,147,483,647	sec
FILEPROCMAX	2,000	3-65535	number
MMAPAREAMAX	40,960	1-16,777,216	page
PROCUSERMAX	25	3-32,767	number
THREADMAX	200	0-100000	number

After you have set individual user limits for users who require higher resource limits, you should consider removing their superuser authority. You should also reevaluate your installation's BPXPRMxx limits and consider reducing them.

For ranges and defaults, see the *z/OS MVS Initialization and Tuning Reference*, SA22-7592.

Surrogate user ID

A surrogate user ID is the identity assigned to client processes when they request PKI Services. A surrogate user ID is required for external clients. For

simplicity, we recommend using surrogate user IDs for internal clients as well, rather than allowing them to access PKI Services under their own identities. Our chosen name for surrogate user ID is PKISRV and the command to create it is:

```
AU PKISRV NAME ('PKI SRVS SURROG') DFLTGRP(MISC) OWNER(MISC) +  
OMVS(AUTOUID HOME('/') PROG('/bin/sh') ASSIZE(256000000) +  
THREADS(512)) NOPASSWORD RESTRICTED
```

We recommend that this user ID be RESTRICTED with the consequences that the user must be explicitly permitted with READ to several resources having UACC(READ) or ID(*) with access READ. The most obvious are:

- ▶ Profile ** in class PROGRAM
- ▶ A few profiles starting with IRR.DIGTCERT in class FACILITY
- ▶ Profile ** in class APPL

(If it has UACC(READ), this equivalent to not having this profile at all).

2.4.3 Adding PKI data set profiles

We recommend that data sets needed for PKI data be named *yourprefix*.PKI, replacing *yourprefix* with a term of your choosing.

Alternatively, if you do not use prefixing, then use an HLQ of PKI.

To define an RACF profile for your PKI data sets, use:

```
AD 'PKI.**' OWNER (PKIADM) AUDIT(ALL)
```

We used PKI as HLQ.

Note: In the command, we assume that you have turned EGN on, and we omitted UACC because we recommend UACC(NONE), which is the default. We believe that by using AUDIT(ALL) your auditors will be happy if SMF records for SUCCESS READ and higher are made available.

The recommended permissions are:

```
PE 'PKI.**' ID(MVSMNT) ACC(ALTER)
```

Usually the MVSMNT group is required to create data sets (VSAM in the case of PKI Services). Do not forget to place your group with ALTER if you plan to create these data sets and you do not have OPERATIONS.

To allow your PKI administrators access to PKI data sets, issue this command:

```
PE 'PKI.**' ID(PKIADM) ACC(CONTROL)
```

Be aware also that the started task user ID for PKI Services (PKISTU) must be able to write into the PKI data sets:

```
PE 'PKI.**' ID(PKISTU) ACC(CONTROL)
```

2.4.4 Using RACF to create certificates

The certificate authority, commonly known as CA, acts as a trusted party to ensure that users who engage in e-business can trust each other. A certificate authority vouches for the identity of each party through the certificates it issues. In addition to proving the identity of the user, each certificate includes a public key that enables the user to verify and encrypt communications.

You can use RACF to create, register, store, and administer digital certificates and their associated private keys, and to build certificate requests that can be sent to a certificate authority for signing. You can also use RACF to manage key rings of stored digital certificates. Digital certificates and key rings are managed in RACF primarily by using the RACDCERT command.

RACF enables you to manage three types of digital certificates:

- User certificate** A certificate that is associated with an RACF user ID and is used to authenticate the user's identity
- CA certificate** A certificate that is associated with a certificate authority and is used to verify signatures in other certificates
- Site certificate** A certificate that is associated with a server or network entity other than a user or certificate authority

Your organization may already have rules for creating certificates. If not, such rules should be discussed and established. An essential part of any certificate is the *distinguished name* that consists of the components listed in Table 2-2.

Table 2-2 Components of Distinguished Name

Common name	(CN)*
Title	(T)*
Organization Unit	(OU)
Organization	(O)
Locality	(L)
Street	(STREET)*
State/Province	(SP)
Mail	(MAIL)*

Postal Code	(POSTALCODE)*
Country	(C)

These abbreviations are used as subkeywords for defining certificates. The fields with * are not typical for CA certificates.

Note: The MAIL, POSTALCODE, and STREET components are valid for PKI Services scripts only. They cannot be used as subkeywords in the RACDCERT command.

The RACDCERT command enables an RACF-defined user ID to use a digital certificate as identification. The certificate must be one of the supported formats contained in an MVS data set. For more information about these formats refer to *z/OS Security Server RACF Command Language Reference, SA22-7687*.

RACDCERT is used to store and maintain digital certificates in RACF and should be used for all maintenance of the DIGTCERT class profiles and related USER profile fields. However, USER-related record type 0207 (User Certificate Name Record) provides user ID, certificate name, and certificate label. For more details see Chapter 10, “RACF Database Unload Utility (IRRDBU00)” in *z/OS V1R4.0 Security Server RACF Macros and Interfaces, SA22-7682*.

The RACDCERT command is your primary administrative tool for managing digital certificates using RACF. Granular authorities for use of the RACDCERT command by users not having SYSTEM SPECIAL are controlled through profiles in the FACILITY class of the type IRR.DIGTCERT.function. It is outside the scope of this book to show the best set of profiles, UACCs, and access lists for these profiles. One possible approach is suggested in 2.4.12, “Protect certificate functions” on page 79, as part of our CLIST to install PKI Services.

The RACDCERT command is used to manage resources in these classes:

- DIGTCERT**

Profiles in the DIGTCERT class contain information about digital certificates as well as the certificate itself. The user ID associated with the certificate can be found in the APPLDATA field of the profile.
- DIGTRING**

Profiles in the DIGTRING class contain information about key rings and the certificates that are part of each key ring. Key rings are named collections of the personal, site, and CA certificates associated with a specific user.
- DIGTNMAP**

Profiles in the DIGTNMAP class contain information about certificate name filters.

Note: Profiles in the DIGTCERT, DIGTRNG, and DIGTNMAP classes are maintained automatically through RADCERT command processing. You cannot administer profiles in these classes using the RDEFINE, RALTER, and RDELETE commands. These commands do not operate with profiles in the DIGTCERT, DIGTRNG, and DIGTNMAP classes.

Profile names in the DIGCERT class are in the form:

serial-number.issuer's-distinguished-name

For example:

41D87A3B05D(serial number)
OU=VeriSign Class1.0=VeriSign.L=Internet (issuer's distinguished name)

Any characters that would not be valid in a profile name, such as a blank, will be replaced with X'4A' (ç). The profile name for this DIGTCERT profile would be:

41D87A3B05.OU=VeriSignçClass1.0=VeriSign.L=Internet

Profile names in the DIGTRNG class are in the form:

owning-userid.key-ring-name

You may use the SEARCH command (SR) or the RLIST command to find all entries defined in the classes. For example:

SR CLASS(DIGTCERT),
SR CLASS(DIGTRNG)

RLIST DIGTCERT * ,
RLIST DIGTRNG * .

The output from RLIST has almost 40 times more lines due to the usual information coming from RLIST, such as OWNER, UACC, date of creation, your access, access list (not applicable here), installation data, and application data.

Create the CA certificate

The command to create this most important certificate is shown in Example 2-15.

Example 2-15 Command to create the certificate authority certificate

```
RADCERT GENCERT CERTAUTH SUBJECTSDN(OU('ITSO PSIE CA')      +  
O('IBM') C('US')) WITHLABEL('IBM ITSO PSIE PKI1')          +  
NOTAFTER(DATE(2020/01/01))
```

GENCERT

Creates a digital certificate and, potentially, a public or private key pair.

CERTAUTH	Relates to certificate of a Certificate Authority (CA)
SUBJECTSDN	Specifies the subject's distinguished name, which consists of the components in Table 2-2 on page 64.

In our case, organization unit is OU=ITSO PSIE CA (ITSO Redbooks in Poughkeepsie, and this is our CA), organization is O=IBM, country is C=US.

WITHLABEL('label-name')

Specifies the label assigned to this certificate. If specified, this must be unique to the user ID with which the certificate is associated.

Note: The label-name is stripped of leading and trailing blanks. If a single quotation mark is intended to be part of the label-name, you must use two single quotation marks together for each single quotation mark within the string, and the entire string must then be enclosed within single quotation marks. We associated this CA certificate with Web server PKI1.

NOTAFTER	Specifies the local date and time after which the certificate is no longer valid.
-----------------	-----------------------------------------------------------------------------------

Note: It is recommended that you set up a job to use the RACDCERT command when a long string of keywords and their values is needed. (See Example 2-16.)

Example 2-16 Job for RACDCERT

```
//your job card
//STEP1      EXEC PGM=IKJEFT01
//SYSLBS     DD DSN=SYS1.BROADCAST,DISP=SHR
//SYSTSPRT   DD SYSOUT=*
//SYSTSIN    DD *
RACDCERT GENCERT CERTAUTH SUBJECTSDN(OU('ITSO PSIE CA') +
O('IBM') C('US')) WITHLABEL('IBM ITSO PSIE PKI1')      +
NOTAFTER(Date(2020/01/01))
```

Making the CA Certificate HIGHTRUST

The subkeywords TRUST NOTRUST HIGHTRUST indicate whether the status of the certificate is trusted, not trusted, or highly trusted. Whether the certificate is not trusted or trusted depends on whether the certificate is valid and whether the private key has been compromised. Because highly trusted certificates are by definition trusted certificates, any certificate usage that was enabled by marking the certificate trusted is also enabled by marking the certificate highly trusted.

However, only CA certificates can be highly trusted. The keyword ALTER is used to change the status to HIGHTRUST:

```
RACDCERT CERTAUTH ALTER(LABEL('IBM ITSO POK PKI1')) HIGHTRUST
```

Note: The value of the LABEL keyword is case-sensitive. (Ours is in uppercase.)

Backup to a data set

It is absolutely critical to back up the certificate into a data set with the PKCS12DER format. The keyword for it is EXPORT:

```
RACDCERT CERTAUTH EXPORT(LABEL('IBM ITSO PSIE PKI1')) +  
DSN('PKI.CAPKI1.BACKUP.P12BIN') +  
FORMAT(PKCS12DER) PASSWORD('xxxxxxx')
```

Note: Be careful when you issue this command because the password is unencrypted, so you have to remember or find a secure place to store the password for future use. If the password is lost, this backup becomes useless. Also the password value is case-sensitive.

Save the CA certificate to a data set for import to a UNIX file

The CA certificate should be placed into an MVS data set in the DER format and then copied to the HFS file. See Example 2-17 for details.

Example 2-17 Placing CA certificate into an MVS data set

```
RACDCERT CERTAUTH EXPORT(LABEL('IBM ITSO POK PKI1')) +  
DSN('PKI.CAPKI1.DERBIN') FORMAT(CERTDER)
```

```
chown PKISTU /var/pkiserv  
oput 'pki.capki1.derbin' '/var/pkiserv/pkiserv1/cacert.der' binary  
chmod 755 /var/pkiserv/cacert.der  
chown pkistu /var/pkiserv/*
```

Create PKI Services key ring

The RACDCERT command to create a key ring for the CA certificate is:

```
RACDCERT ADDRING(CARING) ID(PKISTU),
```

ADDRING(ring name) This creates a key ring. Only users can have a key ring. Key ring names become names of RACF profiles in the DIGTRING class and can contain only characters that are allowed in RACF profile names. Although asterisks are allowed in ring names, a single asterisk is not allowed. For a CA certificate, the user who owns the ring is the PKI daemon. Lowercase characters are permitted. A key ring

name can be up to 237 characters in length. Because only user IDs can have key rings, neither CERTAUTH nor SITE can be specified with ADDRING.

After creating the key ring, the certificate should be added to the key ring. The command is:

```
RACDCERT ID(PKISTU)
CONNECT(CERTAUTH LABEL('IBM ITSO PSIE PKI1') +
RING(CARING) USAGE(PERSONAL) DEFAULT),
```

ID(userid)	Indicates that the certificate being added to the key ring is a user certificate, and userid is the user ID that is associated with this certificate. If the ID keyword is not specified, it defaults to the value specified or the default value on the RACDCERT command.
CONNECT	Specifies that a digital certificate is being added to a key ring.
CERTAUTH	Indicates that the certificate being added to the key ring is a CA certificate.
USAGE	Allows the altering of the trust policy within the confines of a specific key ring.

For example, a CERTAUTH certificate connected with USAGE(PERSONAL) can be used to demote a certificate-authority certificate in order to ensure that it is not used as a certificate authority in this ring. It can be used as a user certificate if a private key is present. Typically, one would not be present.

Consequently, connecting a CERTAUTH certificate as USAGE(PERSONAL) is a way of marking it NOTRUST for this key ring only. Also, a user certificate connected with USAGE(CERTAUTH) can be used to promote an ordinary user certificate to a CA certificate. It can then be used to authenticate user certificates for this key ring only.

DEFAULT	Specifies that the certificate is the default certificate for the ring. Only one certificate within the key ring can be the default certificate. If a default certificate already exists, its DEFAULT status is removed, and the specified certificate becomes the default certificate. If you want the specified certificate to be the default, DEFAULT must be explicitly specified.
---------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Note: All subkeywords belong to CONNECT. (For example, in the RACDCERT command, the left bracket is immediately after CONNECT and the right bracket is after DEFAULT.)

Create the Web server SSL certificate

After we create our CA certificate, we can start issuing other types of certificates, such as an SSL certificate for our Web server started task user ID that is needed to process handshakes with upcoming client certificates belonging to external or internal users.

The command to achieve this is shown in Example 2-18.

Example 2-18 Command to issue the certificate

```
RACDCERT GENCERT ID(WEBSTU) +  
SIGNWITH(CERTAUTH LABEL('IBM ITSO PSIE PKI1')) +  
SUBJECTSDN(CN('wtsc64oe.itso.ibm.com') +  
O('IBM') L('POUGHKEEPSIE') SP('NEW YORK') C('US')) WITHLABEL('SSL PKI1') +  
NOTAFTER(DATE(2020/01/01))
```

```
SIGNWITH(CERTAUTH(LABEL('label-name')))
```

Specifies the certificate with a private key that is signing the certificate. In our case it is our CA certificate. If SIGNWITH is not specified, the default is to sign the certificate with the private key of the certificate that is being generated.

Create the Web server key ring

Similar to the CA, we have to create a key ring for the certificates of our Web server with the command:

```
RACDCERT ADDRING(SSLRING) ID(WEBSTU)
```

We now connect the CA certificate to the ring belonging to the Web server:

```
RACDCERT ID(WEBSTU) CONNECT(CERTAUTH LABEL('IBM ITSO PSIE PKI1') +  
RING(SSLRING))
```

Note: We did not specify USAGE because the default value of USAGE is the same as in the added certificate; That is, we preserved USAGE(CERTAUTH). We also omitted DEFAULT, because we did not want to make the CA certificate be the DEFAULT in this key ring.

We connect the Web server certificate to the Web server ring with this command:

```
RACDCERT ID(WEBSTU) CONNECT(ID(WEBSTU) LABEL('SSL PKI1') RING(SSLRING) +  
USAGE(PERSONAL) DEFAULT)
```

LIST certificates

Now we see what information we get when displaying our certificates.

LIST the CA certificate

```
RACDCERT CERTAUTH LIST(LABEL('IBM ITSO PSIE PKI1'))
```

Note: The keyword CERTAUTH must be used when listing a CA certificate. The value of LABEL is case sensitive, so type IBM and not ibm. Also, CERTAUTH may be placed after list(label(' '))

The output of the command is shown in Table 2-19.

Example 2-19 Listing the CA certificate

Digital certificate information for CERTAUTH:

```
Label: IBM ITSO PSIE PKI1
Certificate ID: 2QiJmZmDhZmjgcnC1EDJ4+LWQNfiycVA19LJ8UBA
Status: HIGHTRUST
Start Date: 2003/04/30 00:00:00
End Date: 2020/01/01 23:59:59
Serial Number:
>00<
Issuer's Name:
>OU=ITSO PSIE CA.O=IBM.C=US<
Subject's Name:
>OU=ITSO PSIE CA.O=IBM.C=US<
Key Usage: CERTSIGN
Private Key Type: ICSF
Private Key Size: 1024
Ring Associations:
Ring Owner: PKISTU
Ring:
>CARING<
Ring Owner: WEBSTU
Ring:
>SSLRING<
```

Now we RLIST our CA certificate:

```
RL DIGTCERT 00.OU=ITSO¢PSIE¢CA.O=IBM.C=US ALL
```

The output is shown in Example 2-20.

Example 2-20 RLIST of our certificate

CLASS	NAME
-----	----
DIGTCERT	00.OU=ITSO¢PSIE¢CA.O=IBM.C=US

LEVEL	OWNER	UNIVERSAL ACCESS	YOUR ACCESS	WARNING
-------	-------	------------------	-------------	---------

```

-----
00      ANTOFF      ALTER      ALTER      NO

INSTALLATION DATA
-----
NONE

APPLICATION DATA
-----
irrcerta

SECLEVEL
-----
NO SECLEVEL

CATEGORIES
-----
NO CATEGORIES

SECLABEL
-----
NO SECLABEL

AUDITING
-----
FAILURES(READ)

GLOBALAUDIT
-----
NONE

NOTIFY
-----
NO USER TO BE NOTIFIED
CREATION DATE  LAST REFERENCE DATE  LAST CHANGE DATE
(DAY) (YEAR)      (DAY) (YEAR)      (DAY) (YEAR)
-----
120  03          120  03          120  03

ALTER COUNT  CONTROL COUNT  UPDATE COUNT  READ COUNT
-----
000000      000000      000000      000000

USER      ACCESS  ACCESS COUNT
-----
NO USERS IN ACCESS LIST

ID      ACCESS  ACCESS COUNT  CLASS      ENTITY  NAME
-----

```

Note: The value of the UACC denotes the status of the certificate: ALTER if the certificate is TRUSTed or HIGHTRUSTed. The APPLDATA field contains the user ID associated with the certificate. (In our CA certificate, it is the IBM-supplied user ID irrcerta. User ID irrcerta will be always in APPLDATA for NOTRUSTed certificates.

LIST the Web server (SSL) certificate

The command to display the Web server certificate is:

```
RACDCERT ID(WEBSTU) LIST(LABEL('SSL PKI1'))
```

The output from this command is shown in Table 2-21.

Note: The keyword ID() must be used when listing a user certificate. Also, ID() may be placed after list(label(' ')). Remember that the value of LABEL is case sensitive.

Example 2-21 Listing certificate for Web server

Digital certificate information for user WEBSTU:

```
Label: SSL PKI1
Certificate ID: 2QbmxCLi4+Ti4tNA19LJ8UBA
Status: TRUST
Start Date: 2003/04/30 00:00:00
End Date: 2020/01/01 23:59:59
Serial Number:
>00<
Issuer's Name:
>OU=ITSO PSIE CA.0=IBM.C=US<
Subject's Name:
>CN=wtsc64oe.itso.ibm.com.0=IBM.L=POUGHKEEPSIE.SP=NEW YORK.C=US<
Private Key Type: Non-ICSF
Private Key Size: 1024
Ring Associations:
Ring Owner: WEBSTU
Ring:
>SSLRING<
```

Now, we list our Web server certificate issued by our CA:

```
RL DIGTCERT 01.OU=ITSO¢PSIE¢CA.0=IBM.C=US ALL
```

Part of the output is shown in Example 2-22 on page 74.

Example 2-22 RLIST of the Web server certificate

```
CLASS      NAME
-----
DIGTCERT   01.OU=ITS0¢PSIE¢CA.0=IBM.C=US

LEVEL  OWNER      UNIVERSAL ACCESS  YOUR ACCESS  WARNING
-----
00     ANTOFF             ALTER             ALTER         NO

INSTALLATION DATA
-----
NONE

APPLICATION DATA
-----
WEBSTU

-----
-----
```

Note: The only difference between the profile for our Web server certificate and the profile for our CA certificate is the serial number. The APPLDATA field contains the user ID associated with certificate: the Web server started task user ID: WEBSTU.

LIST key ring

Now we list our key rings.

For user ID PKISTU

The command to list the key ring is:

```
RACDCERT ID(PKISTU) LISTRING(CARING)
```

Table 2-23 shows the output.

Note: The keyword ID() must always be used to list any key ring. Also ID() may be placed after LISTRING(). Again, the key ring name is case sensitive.

Example 2-23 List of the key ring for PKISTU

Digital ring information for user PKISTU:

```
Ring:
>CARING<
Certificate Label Name          Cert Owner  USAGE      DEFAULT
```


-----	-----	-----	-----
IBM ITSO PSIE PKI1	CERTAUTH	PERSONAL	YES

The output from RLIST PKISTU.CARING ALL is shown in Example 2-24.

Example 2-24 Output of RLIST PKISTU

CLASS	NAME			
-----	----			
DIGTRING	PKISTU.CARING			
LEVEL	OWNER	UNIVERSAL ACCESS	YOUR ACCESS	WARNING
-----	-----	-----	-----	-----
00	ANTOFF	NONE	NONE	NO
INSTALLATION DATA				

NONE				
APPLICATION DATA				

PKISTU				
SECLEVEL				

NO SECLEVEL				
CATEGORIES				

NO CATEGORIES				
SECLABEL				

NO SECLABEL				
AUDITING				

FAILURES(READ)				
GLOBALAUDIT				

NONE				
NOTIFY				

NO USER TO BE NOTIFIED				
CREATION DATE	LAST REFERENCE DATE		LAST CHANGE DATE	
(DAY) (YEAR)	(DAY) (YEAR)		(DAY) (YEAR)	
-----	-----		-----	

120	03	120	03	120	03
ALTER COUNT		CONTROL COUNT		UPDATE COUNT	
-----		-----		-----	
000000		000000		000000	
USER		ACCESS		ACCESS COUNT	
----		-----		-----	
NO USERS IN ACCESS LIST					
ID	ACCESS	ACCESS COUNT	CLASS	ENTITY	NAME
-----		-----		-----	
NO ENTRIES IN CONDITIONAL ACCESS LIST					

For user ID WEBSTU

Example 2-25 Listing key ring of Web server

Digital ring information for user WEBSTU:

Ring:				
>SSLRING<				
Certificate Label	Name	Cert Owner	USAGE	DEFAULT
-----	-----	-----	-----	-----
IBM ITSO PSIE PKI1		CERTAUTH	PERSONAL	NO
SSL PKI1		ID(WEBSTU)	PERSONAL	YES

2.4.5 Daemon and server control for PKI user ID and surrogate user ID

We permit PKISTU read access to BPX.DAEMON and BPX.SERVER in the RACF class FACILITY, and PKISRV read access to BPX.SERVER in the RACF class FACILITY:

```
PE BPX.DAEMON CL(FACILITY) ID(PKISTU) ACC(R)
PE BPX.SERVER CL(FACILITY) ID(PKISTU) ACC(R)
PE BPX.SERVER CL(FACILITY) ID(PKISRV) ACC(R)
```

2.4.6 Allow PKI user ID to act as CA

The following commands enable a PKI user ID to act as a CA:

```
RDEF FACILITY IRR.DIGTCERT.GENCERT UACC(NONE) OWNER(PKIADM)+
AUDIT(ALL)
RDEF FACILITY IRR.DIGTCERT.LIST UACC(NONE) OWNER(PKIADM)
RDEF FACILITY IRR.DIGTCERT.LISTRING UACC(NONE) OWNER(PKIADM)
PE IRR.DIGTCERT.GENCERT CL(FACILITY) ID(PKISTU) ACC(CONTROL)
PE IRR.DIGTCERT.LIST CL(FACILITY) ID(PKISTU) ACC(R)
PE IRR.DIGTCERT.LISTRING CL(FACILITY) ID(PKISTU) ACC(R)
```

Note: User IDs having SPECIAL can issue the RACDCERT command with all keywords and parameters. For a list of all possible profiles in the FACILITY class for protection of the RACDCERT command refer to Chapter 20 in the *Security Server RACF Security Administration Guide, SA22-7683*.

2.4.7 Allow Web server to access its own key ring

```
PE IRR.DIGTCERT.LIST CL(FACILITY) ID(WEBSTU) ACC(R)
PE IRR.DIGTCERT.LISTRING CL(FACILITY) ID(WEBSTU) ACC(R)
```

2.4.8 Allow Web server user ID to switch identity to surrogate user ID

The commands to enable the Web server user ID to act on behalf of the surrogate user ID PKISRV (or switch identity to surrogate user ID) are:

```
RDEF SURROGAT BPX.SRV.PKISRV
PE BPX.SRV.PKISRV CL(SURROGAT) ID(WEBSTU) ACC(R)
```

2.4.9 Profile for PKI Services procedure in class STARTED

```
RDEF STARTED PKISRV*.** STDATA(USER(PKISTU) GROUP(STG)) OWNER(PKIADM)
```

This covers more than one PKI server (for example, if you want a few members in SYS1.PROCLIB named PKISRV1, PKISRV2, and so on). However, you can run only one PKI server per LPAR at a time.

2.4.10 Allow access for PKISTU to OCSF

For the PKI Services procedure to start, the PKI Services user ID must have access to the OCSF services. This access is provided by issuing this command:

```
PE CDS.** CL(FACILITY) ID(PKISTU) ACC(R)
```

2.4.11 ICSF

Here we assume that ICSF is active. To store a certificate into ICSF, use:

```
RACDCERT CERTAUTH ADD('PKI.CAPKI1.BACKUP.P12BIN')          +
PASSWORD('xxxxxxx') ICSF
```

ADD(*data-set-name*) Specifies that a digital certificate is to be defined. The specified data set must contain the digital certificate. The data set containing the digital certificate or certificate package must be cataloged and cannot be a PDS or a PDS member. The RECFM expected by RACDCERT is

VB. When the ADD keyword is specified, RACDCERT dynamically allocates and opens the specified data set and reads the certificate from it as binary data.

Note: The issuer of the RACDCERT command must have READ access to the *data-set-name* data set to prevent an authorization abend from occurring when the data set is read. Each user ID can be associated with more than one digital certificate but they must be added individually. The specified data set should contain only one digital certificate. The command reads the certificate from the data set, updates the user's profile, and creates the DIGTCERT profile.

PASSWORD('pkcs12-password')

Specifies the password that is associated with the PKCS#12 certificate package. This keyword is required if the data set is PKCS#12, and it must not be specified if the data set is not PKCS#12. The 'pkcs12-password' can be up to 255 characters in length, is case sensitive, and can contain blanks.

Note: The specified password will be visible on the screen, so care should be taken to prevent it from being viewed when entered. Because PKCS#12 passwords do not follow the normal TSO/E rules for password content, they cannot be suppressed as they normally would be.

ICSF

ICSF specifies that RACF should attempt to store the private key associated with this certificate in the ICSF PKDS. This applies when the key is introduced to RACF by issuing the ADD keyword for PKCS#12 certificate packages and when an existing certificate profile containing a non-ICSF private key is replaced by issuing the ADD keyword.

If the GENCERT keyword creates a public/private key pair and ICSF is being used to store private keys, then GENCERT creates an ICSF key label in the format *IRR.DIGTCERT.userid.cvtsname.ebcdic-stck-value*, where *userid* is the owning user ID, *CVTSNAME* is the system name as taken from the CVT, and *ebcdic-stck-value* is an EBCDIC version of the current store clock value. If the key is associated with a CA certificate, then *user ID* is set to CERTIFAUTH. If the key is associated with a site certificate, then *user ID* is set to SITECERTIF.

Ensure that you have created a profile in class CSFKEYS as follows:

```
RDEF CSFKEYS IRR.DIGTCERT.** OWNER(PKIADM)
PE IRR.DIGTCERT.** CL(CSFKEYS) ID(PKIADM) ACC(R)
```

Note: This is the violation message if you were not authorized to profile IRR.DIGTCERT.** in class CSFKEYS:

```
ICH408I USER(ANTOFF ) GROUP(SYS1 ) NAME(THEO ANTOFF)
IRR.DIGTCERT.CERTIFAUTH.SC64.B953A8B607F4EC81
CL(CSFKEYS )
INSUFFICIENT ACCESS AUTHORITY
FROM IRR.DIGTCERT.CERTIFAUTH.** (G)
ACCESS INTENT(READ ) ACCESS ALLOWED(NONE)
```

2.4.12 Protect certificate functions

A list of all PKI functions (end user and administrative) can be found in the *Security Server RACF Security Administration Guide*, SA22-7683, and in 2.5.7, “Controlling applications that call R_PKIServ” on page 94.

You have to allow your surrogate PKI user ID to use end-user certificate functions with the commands:

```
RDEF FACILITY IRR.RPKISERV.** OWNER(PKIADM)
PE IRR.RPKISERV.** CL(FACILITY) ID(PKISRV) ACC(CONTROL)
```

Note: By default (and we are using the default) in the pkiserv.tmpl file, four of the eight types of PKI certificates are auto-approved (the two SAF certificates, the 2-Year PKI Browser Certificate for Authenticating To z/OS, and the 5-Year PKI Intermediate CA certificate). The reason to give CONTROL to PKISRV for the end-user functions of the PKI callable service is to cater to those auto-approvals not requiring administrator action.

A single profile IRR.RPKISERV.PKIADMIN in class FACILITY can protect all six administrative-user functions. (For a description of these functions, see 2.5.7, “Controlling applications that call R_PKIServ” on page 94). To disallow access to the administrative-user functions of PKI Services for the surrogate PKI user ID, place PKISRV on its access list with NONE:

```
RDEF FACILITY IRR.RPKISERV.PKIADMIN OWNER(PKIADM)
PE IRR.RPKISERV.PKIADMIN ID(PKISRV) ACC(NONE)
```

Note: The reason for explicit access NONE for PKISRV is that we placed this user ID on the access list of IRR.RPKISERV.** covering both end-user and administrative functions. Discrete profile IRR.RPKISERV.PKIADMIN covers all administrative functions, but it is independent of the preceding generic profile, not a more specific profile derived from it.

We assume that all members of your PKIADM group have SYSTEM SPECIAL. If this is not the case, then members without SYSTEM SPECIAL must have UPDATE to profile IRR.RPKISERV.PKIADMIN.

This was the last command needed to define the RACF environment for PKI Services. Now, with this background, you may wish to use one of three almost equivalent options to execute the commands:

1. Run the REXX member IKYSETUP from SYS1.SAMPLIB after copying it to your own library and updating with your site values according to the tables in Chapter 4 of *z/OS Security Server PKI Services Guide and Reference*, SA22-7693.
2. Run the CLIST shown in Example 2-26 as a JCL job. It is a stack of RACF commands explained in 2.4, “Setting up the RACF environment for PKI Services” on page 61 after you have carefully checked that you have already all the controls discussed and explained in 2.3, “Setting up RACF environment for PKI prerequisites” on page 55.
3. Execute the commands included in section 2.4, “Setting up the RACF environment for PKI Services” on page 61 as you progress reading it. Do not forget refreshes for RACLSTED or GENERIC classes.

Example 2-26 Plug RACF environment for PKI Services

```
PROC 0
/*****
/*          RACF ENVIRONMENT FOR PKI SERVICES          */
/*          */
/* A sample CLIST to execute all RACF commands for installing */
/*          PKI Services          */
/* You must change this CLIST to satisfy your own shop standards */
/* Comment out commands which you may decide not to run */
/* Prereq 1: run CLIST RACF ENVIRONMENT FOR PREREQS FOR PKI SERVICES */
/* Prereq 2: all members of group PKIADM have SYSTEM SPECIAL */
/* If you do not want your PKI Admin to have SYSTEM SPECIAL than give */
/* them access CONTROL to all profiles IRR.DIGTCERT in class FACILITY */
/* Maintained by: your RACF/PKI administrators */
/* History: */
/* date */
/* */
/*****
/*          */
/*  G R O U P S          */
/*          */
/*****
    AG PKIADM SUPGROUP(JOBROLE) OWNER(JOBROLE) OMVS(AUTOGID)
    CO (user1 user2 etc userids of your PKI administrators) GROUP(PKIADM)
    AG STG SUPGROUP(DFLT) OWNNER(DFLT) OMVS(AUTOGID)
```

```

/* ALG (your RACF admin group) OMVS(AUTOUID) */
/* ALG (your default group for started task usersids) GID(AUTOUID) */
/*****
/*
/*
/*   U S E R I D S
/*
/*****
    AU PKISTU NAME('PKI SRVS DAEMON') DFLTGRP(STG) OWNER(STG) +
    OMVS(AUTOUID HOME('/') PROG('/bin/sh') ASSIZE(256000000) +
    THREADS(512)) NOPASSWORD
    AU PKISRV NAME ('PKI SRVS SURROG') DFLTGRP(MISC) OWNER(MISC) +
    OMVS(AUTOUID HOME('/') PROG('/bin/sh') ASSIZE(256000000) +
    THREADS(512)) NOPASSWORD RESTRICTED
/*****
/*
/*   PERMITS DUE TO RESTRICTED ATTRIBUTE FOR PKISRV */
/*
/*****
PE IRR.DIGTCERT.** CL(FACILITY) ID(PKISRV) ACC(R)
    PE IRR.DIGTCERT.ADD CLASS(FACILITY) ID(PKISRV) ACC(R)
    PE IRR.DIGTCERT.ADDRING CLASS(FACILITY) ID(PKISRV) ACC(R)
    PE IRR.DIGTCERT.CONNECT CLASS(FACILITY) ID(PKISRV) ACC(R)
    PE IRR.DIGTCERT.DELETE CLASS(FACILITY) ID(PKISRV) ACC(R)
    PE IRR.DIGTCERT.DELRING CLASS(FACILITY) ID(PKISRV) ACC(R)
    PE IRR.DIGTCERT.LIST CLASS(FACILITY) ID(PKISRV) ACC(R)
    PE IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(PKISRV) ACC(R)
    PE IRR.DIGTCERT.REMOVE CLASS(FACILITY) ID(PKISRV) ACC(R)
PE ** CL(PROGRAM) ID(PKISRV) ACC(R)
/*PE ** CL(APPL) ID(PKISRV) ACC(R)
/*****
/*
/*
/*           D A T A S E T S
/*
/*
/*****
    AD 'PKI.**' OWNER(PKIADM) AUDIT(ALL)
    PE 'PKI.**' ID(PKIADM) ACC(ALTER)
    PE 'PKI.**' ID(PKISTU) ACC(CONTROL)
/*****
/*
/*
/*           C A   C E R T I F I C A T E
/*
/*
/*****
    RACDCERT GENCERT CERTAUTH SUBJECTSDN(OU('ITSO PSIE CA')
    O('IBM') C('US')) WITHLABEL('IBM ITSO PSIE PKI1')
    NOTAFTER(DATE(2020/01/01))
    RACDCERT CERTAUTH ALTER(LABEL('IBM ITSO PSIE PKI1')) HIGHTRUST
    RACDCERT CERTAUTH EXPORT(LABEL('IBM ITSO PSIE PKI1'))
    DSN('PKI.CAPKI1.BACKUP.P12BIN') FORMAT(PKCS12DER)
    PASSWORD('xxxxxxx')

```

```

RACDCERT CERTAUTH EXPORT(LABEL('IBM ITSO PSIE PKI1'))          +
DSN('PKI.CAPKI1.DERBIN') FORMAT(CERTDER)
RACDCERT CERTAUTH ADD('PKI.CAPKI1.BACKUP.P12BIN')              +
PASSWORD('xxxxxxx') ICSF
/*****/
/*
/*          P K I   S E R V I C E S   K E Y   R I N G          */
/*
/*****/
RACDCERT ADDRING(CARING) ID(PKISTU)
RACDCERT ID(PKISTU) CONNECT(CERTAUTH LABEL('IBM ITSO PSIE PKI1') +
RING(CARING) USAGE(PERSONAL) DEFAULT)
/*****/
/*
/*          W E B   S E R V E R   C E R T I F I C A T E   ( S S L )
/*
/*
/*****/
RACDCERT GENCERT ID(WEBSTU)                                     +
SIGNWITH(CERTAUTH LABEL('IBM ITSO PSIE PKI1'))                +
SUBJECTSDN(CN('wtsc64oe.itso.ibm.com') O('IBM'))              +
L('Poughkeepsie') SP('New York') C('US'))                    +
WITHLABEL('SSL PKI1') NOTAFTER(DATE(2020/01/01))
/*****/
/*
/*          W E B   S E R V E R   K E Y   R I N G          */
/*
/*
/*****/
RACDCERT ADDRING(SSLRING) ID(WEBSTU)
RACDCERT ID(WEBSTU) CONNECT(CERTAUTH LABEL('IBM ITSO PSIE PKI1') +
RING(SSLRING))
RACDCERT ID(WEBSTU) CONNECT(ID(WEBSTU)                          +
LABEL('SSL PKI1') RING(SSLRING) USAGE(PERSONAL) DEFAULT)
/*****/
/*
/*          P R O F I L E   I N   C L A S S   S U R R O G A T   F O R   S U R R O G A T E   U S E R I D
/*
/*
/*****/
RDEF SURROGAT BPX.SRV.PKISRV OWNER(PKIADM)
PE BPX.SRV.PKISRV CL(SURROGAT) ID(WEBSTU) ACC(R)
/*****/
/*
/*          D A E M O N   A N D   S E R V E R   C O N T R O L
/*
/*
/*****/
PE BPX.DAEMON CL(FACILITY) ID(PKISTU) ACC(R)
PE BPX.SERVER CL(FACILITY) ID(PKISTU) ACC(R)
PE BPX.SERVER CL(FACILITY) ID(PKISRV) ACC(R)
/*****/
/*

```



```

/*          CERTIFICATE ADMINISTRATION          */
/*
/*****
RDEF FACILITY IRR.DIGTCERT.** OWNER(PKIADM)
RDEF FACILITY IRR.DIGTCERT.ADD OWNER(PKIADM))
RDEF FACILITY IRR.DIGTCERT.ADDRING OWNER(PKIADM)
RDEF FACILITY IRR.DIGTCERT.CONNECT OWNER(PKIADM)
RDEF FACILITY IRR.DIGTCERT.DELETE OWNER(PKIADM)
RDEF FACILITY IRR.DIGTCERT.DELRING OWNER(PKIADM)
RDEF FACILITY IRR.DIGTCERT.GENCERT OWNER(PKIADM)
RDEF FACILITY IRR.DIGTCERT.LIST OWNER(PKIADM)
RDEF FACILITY IRR.DIGTCERT.LISTRING OWNER(PKIADM)
RDEF FACILITY IRR.DIGTCERT.REMOVE OWNER(PKIADM)
PE IRR.DIGTCERT.** CL(FACILITY) ID(PKISTU PKISRV) ACC(C)
PE IRR.DIGTCERT.ADD CL(FACILITY) ID(*) ACC(R)
PE IRR.DIGTCERT.ADD CL(FACILITY) ID(PKISTU PKISRV) ACC(U)
PE IRR.DIGTCERT.ADDRING CL(FACILITY) ID(*) ACC(R)
PE IRR.DIGTCERT.CONNECT CL(FACILITY) ID(*) ACC(R)
PE IRR.DIGTCERT.DELETE CL(FACILITY) ID(*) ACC(R)
PE IRR.DIGTCERT.DELRING CL(FACILITY) ID(*) ACC(R)
PE IRR.DIGTCERT.GENCERT CL(FACILITY) ID(PKISTU PKISRV) ACC(C)
PE IRR.DIGTCERT.LIST CL(FACILITY) ID(*) ACC(R)
PE IRR.DIGTCERT.LIST CL(FACILITY) ID(PKISTU PKISRV) ACC(C)
PE IRR.DIGTCERT.LISTRING CL(FACILITY) ID(*) ACC(R)
PE IRR.DIGTCERT.LISTRING CL(FACILITY) ID(PKISTU PKISRV) ACC(C)
PE IRR.DIGTCERT.REMOVE CL(FACILITY) ID(*) ACC(R)
RDEF FACILITY IRR.RPKISERV.** OWNER(PKIADM)
RDEF FACILITY IRR.RPKISERV.PKIADMIN OWNER(PKIADM)
PE IRR.RPKISERV.** CL(FACILITY) ID(PKISRV) ACC(C)
PE IRR.RPKISERV.PKIADMIN CL(FACILITY) ACC(NONE)
*****/
/*
/*          C L A S S   S T A R T E D          */
/*
/*****
RDEF STARTED PKISRV*.** STDATA(USER(PKISTU) GROUP(STG)) +
OWNER(PKIADM)
/*****
/*
/*          ACCESS TO OCSF          */
/*
/*****
PE CDS.** CL(FACILITY) ID(PKISTU) ACC(R)
/*****
/*
/*          C L A S S E S   C S F K E Y S   A N D   C S F S E R V          */
/*
/*****
RDEF CSFKEYS IRR.DIGTCERT.** OWNER(PKIADM)

```

```

      PE IRR.DIGTCERT.** CL(CSFKEYS) ID(PKISTU PKIADM) ACC(R)
      PE CSF* CL(CSFSERV) ID(PKISRV PKIADM LDAPSTU PKISTU) ACC(R)
      /*****
      /*
      /*          S E T R   R E F R E S H
      /*
      /*
      /*****
      SETR RACLIST(CSFSERV) REFR
      SETR RACLIST(CSFKEYS) REFR
      SETR GENERIC(DATASET) REFR
      SETR RACLIST(FACILITY) REFR
      SETR RACLIST(STARTED) REFR
      SETR RACLIST(SURROGAT) REFR
      SETR WHEN(PROGRAM) REFR
      /*****

```

2.5 RACF administration for PKI Services

This section describes the possible activity by RACF administrators after PKI Services has been set up and customized.

2.5.1 Creating a help desk function

Usually your help desk group must have inquiry ability in the frame of PKI Services. The commands for this scenario are shown in Example 2-27.

Example 2-27 Commands for help desk inquire ability

```

AG HELPDESK SUPGROUP(JOBROLE) OWNER(JOBROLE) + OMVS(AUTOGID)
AU HDUSR1 DFLTGRP(EMPL) OWNER(EMPL) OMVS(AUTOUID)
CO HDUSR1 GROUP(HELPDESK)
PE 'PKI.**' ID(HELPDESK) ACC(R)
PE IRR.RPKISERV.PKIADMIN CL(FACILITY) ID(HELPDESK) ACC(R)
PE IRR.DIGTCERT.LIST CL(FACILITY) ID(HELPDESK) ACC(C)
PE IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(HELPDESK) ACC(C)

```

Access CONTROL to the last two profiles ensures that help desk members can list anybody's certificates or key rings by issuing the following commands.

For any user ID having certificates registered in RACF:

```
RACDCERT ID(userid) LIST
```

If they wish to find all CA certificates:

```
RACDCERT CERTAUTH LIST
```

On the other hand, the last two profiles in Example 2-27 on page 84 must have UACC(READ) or ID(*) on their access list with READ. Thus every user ID will be able to display their certificates or key rings by issuing:

RACDCERT LIST Obtains detailed information for each certificate

RACDCERT LISTRING(*) Obtains detailed information for each key ring

Note: Help desk can use RACDCERT ID(userid) LISTRING(*) to find all rings associated with a user ID, but a similar command RACDCERT LIST(*) fails with a message:

```
IKJ56712I INVALID KEYWORD, *  
IKJ56703A REENTER THIS OPERAND - .
```

2.5.2 Administering certificates with the HostIdMappings extension

You can add a HostIdMappings extension to certificates you create for certain users, enabling you to specify the user IDs for logging on to particular servers (or hosts). Controlling an identity used for logon purposes is a very important security objective. Therefore, you must exercise administrative control in the following areas by authorizing:

- ▶ PKI Services as a highly trusted certificate authority whose certificates will be honored when they contain HostIdMappings extensions
- ▶ Particular servers to accept logons from clients whose certificates contain HostIdMappings extensions

To enable the Web server to accept logons from clients who have been issued PKIServices certificates with HostIdMapping extensions, you must create profiles in the RACF class SERVAUTH.

Note: Ensure that your CA certificate is altered to HIGHTRUST, if it was not when you created it. See “Making the CA Certificate HIGHTRUST” on page 67.

Define a profile in class SERVAUTH for each server (host) name you want your Web server to honor when accepting logons for certificates containing HostIdMapping extensions. The profile has the format IRR.HOST.*hostname* where the value of *hostname* usually is a domain name, such as wtsc63oe.itso.ibm.com. This domain name must be entered in the entry for HostIdMap in /web/pki1/pkiserv/pkiserv.tmpl but without the subject ID portion in the APPL section.

```
RDEF SERVAUTH IRR.HOST.WTSC63OE.ITSO.IBM.COM UACC(NONE)
```

Permit your Web server started task user ID with READ:

```
PE IRR.HOST.WTSC630E.ITS0.IBM.COM CL(SERVAUTH) ID(WEBSTU) ACC(R)
```

Now activate and raclist class SERVAUTH:

```
SETR CLASSACT(SERVAUTH)
SETR RACLIST(SERVAUTH)
SETR RACLIST(SETRAUTH) REFR
```

Note: On a z/OS system, a HostIdMapping is not honored if the target user ID was created after the start of the validity period for the certificate containing the HostIdMappings extension. Therefore, if you are creating user IDs specifically for certificates with HostIdMappings extensions, ensure that you create the user IDs before the certificate requests are submitted.

2.5.3 Display your PKI Services certificates

Often you may wish to display your CA certificate or key ring, possibly to diagnose error conditions.

Display CA certificate

You can do this by using two RACF commands. The first one is:

```
RACDCERT CHECKCERT('PKI.CAPKI1.DERBIN')
```

In this command, PKI.CAPKI1.DERBIN is the data set to which we saved the CA certificate in DER format. The output of the first RACF command is shown in Example 2-28.

Example 2-28 RACDCERT CHECKCERT command output

Digital certificate information for CERTAUTH:

```
Label: IBM ITS0 PSIE PKI1
Certificate ID: 2QiJmZmDhZmjgcnC1EDJ4+LWQNfiycVA19LJ8UBA
Status: HIGHTRUST
Start Date: 2003/04/25 00:00:00
End Date: 2020/01/01 23:59:59
Serial Number:
>00<
Issuer's Name:
>OU=ITS0 PSIE CA.O=IBM.C=US<
Subject's Name:
>OU=ITS0 PSIE CA.O=IBM.C=US<
Key Usage: CERTSIGN
Private Key Type: ICSF
Private Key Size: 1024
```

It is important to note, when diagnosing errors, that:

- ▶ The first line must indicate that this is a CERTAUTH certificate.
- ▶ The Private Key Type and Size must be present.
- ▶ If the Serial Number is not equal to 00, this indicates that the certificate has been renewed or was issued by another certificate authority.

The second RACF command to display the CA certificate uses the LIST keyword of RACDCERT:

```
RACDCERT CERTAUTH LIST(LABEL('ITSO IBM PSIE PKI1'))
```

Its output is the same as the first, with ring associations added (Example 2-29).

Example 2-29 RACDCERT CERAUTH LIST command output

```
Ring Associations:
  Ring Owner: PKISTU
  Ring:
    >CARING<
  Ring Owner: WEBSTU
  Ring:
    >SSLRING<
```

From this information, you should ensure that one of the associations listed has the PKI Services daemon user ID as the ring owner and that the ring name matches your CA key ring name as listed in example Example 2-30.

Display CA key ring

To display the CA key ring, enter the following command:

```
RACDCERT ID(PKISTU) LISTRING(CARING)
```

Example 2-30 shows the output.

Example 2-30 Display CA key ring for CA

Digital ring information for user PKISTU:

Ring:			
>CARING<			
Certificate Label Name	Cert Owner	USAGE	DEFAULT
-----	-----	-----	-----
IBM ITSO PSIE PKI1	CERTAUTH	PERSONAL	YES

The ring information must have USAGE = PERSONAL and DEFAULT = YES.

Display certificates using utilities iclview and vosview

Sometimes you may wish to display your Issued Certificates List (ICL) or your Certificates Request List (CRL). These lists are kept respectively in VSAM data sets PKI.WEBPKI1.ICL and PKI.WEBPKI1.OST. These lists are independent of RACF, although ICL may contain certificates registered in RACF. To display ICL and CRL, two utilities can be used: iclview and vosview, respectively. Issue the OMVS command and do the following to display all certificates issued by PKI Services:

```
cd /usr/lpp/pkiserv/lib
/usr/lpp/pkiserv/bin/iclview/ \`pki.webpki1.icl\`
```

The output from iclview utility is shown in Example 2-31.

Example 2-31 ICL utility output

Cert 1:

ISSUED (CA's certificate)
Issued at 2003-04-30 17:59:57
Last changed 2003-04-30 17:59:57
Subject: OU=ITSO PSIE CA,O=IBM,C=US
Issuer: OU=ITSO PSIE CA,O=IBM,C=US
Requester:
AppData:
Serial Number: 0
Email flag: Off

Cert 2: Theo Antoff

ISSUED (Issued certificate)
Issued at 2003-05-15 16:30:05
Last changed 2003-05-15 16:30:05
Subject: CN=Theo Antoff,OU=ITSO Class 1 Internet Certificate CA,O=IBM,C=US
Issuer: OU=ITSO PSIE CA,O=IBM,C=US
Requester: Theo Antoff
AppData: 1YBSSL
Serial Number: 15
Email flag: Off

To display all certificate requests received by PKI Services after issuing OMVS:

```
cd /usr/lpp/pkiserv/lib
/usr/lpp/pkiserv/bin/vosview/ \`pki.webpki1.ost\`
```

The output from the vosview utility is shown in Example 2-32.

Example 2-32 Displaying all certificate requests received by PKI Services

Object key = 1
Last used key = 108, CRL serial number = 4, ARL serial number = 4

```
High DP = 0, Low DP = 1
name = ""
tid = ??????????????????????
apldata =
comment =
data len = 20
flags = 0 - Type = ??? ObjSt?????
Creation time is: 2003/05/16 13:49:06
Last modified time is: 2003/05/19 12:05:55
```

```
Object key = 2
name = ""
tid = ??????????????????????
apldata =
comment =
data len = 33
flags = 0 - Type = ??? ObjSt?????
Creation time is: 2003/05/16 13:49:08
Last modified time is: 2003/05/18 20:49:15
```

```
Object key = 108
name = "ta"
tid = 1jJVzyGR0Uwe2Qn07++++++
apldata = 1YBSSL
comment =
data len = 1116
flags = 1040010 - Type = Cert State = RA CertReqActive
Creation time is: 2003/05/19 12:05:55
Last modified time is: 2003/05/19 12:05:55
```

Records starting with Object key = 1 and Object key =2 contain only system data.

Records starting with Object key=108 is a request sent by user ta for a 1-Year PKI SSL Browser certificate. (The type of certificate is represented by apldata =1YBSSL). For more information about the meaning of the other fields in the records, refer to Chapter 20 of *z/OS Security Server PKI Services Guide and Reference*, SA22-7693.

2.5.4 Establishing PKI Services as intermediate certificate authority

The default setup for PKI Services establishes the PKI Services certificate authority as a root CA, also known as a self-signed CA. Because there is no established trust hierarchy leading to a self-signed certificate, it is impossible to verify that a self-signed certificate is genuine. Accordingly, any person or

application that wishes to process certificates issued by a root authority must explicitly trust the authenticity of the self-signed CA certificate.

Alternately, you can establish the PKI Services certificate authority as an intermediate (subordinate) certificate authority. An intermediate certificate authority is one whose certificate is signed by another higher certificate authority.

Perform the following steps to establish PKI Services as an intermediate certificate authority:

1. Determine which certificate authority will be acting as a higher authority for your PKI Services. (This could be a public CA such as VeriSign).
2. Create a new certificate request from your self-signed CA certificate by entering the following RACF command:

```
RACDCERT CERTAUTH GENREQ(LABEL('IBM ITSO PSIE PKI1')) +  
DSN('PKI.CAPKI1.BACKUP.P12BIN')
```

3. Send the certificate request to the higher certificate authority, following its required procedures.
4. After the certificate has been issued, receive the certificate back into the certificate data set (PKI.CAPKI1.BACKUP.P12BIN).

Note: The procedure for doing this can vary greatly depending on how the higher certificate authority delivers the new certificate: If the certificate is delivered as base64 encoded text, the easiest way to deposit the certificate into the data set is to copy and paste the certificate into the empty data set. If the certificate is delivered as binary data (also called DER-encoded), the easiest way to deposit the certificate into the data set is to use binary FTP.

5. Receive the certificate back into the RACF database by entering the following RACF command:

```
RACDCERT CERTAUTH ADD('PKI.CAPKI1.BACKUP.P12BIN')
```

6. Export the certificate in DER format to the export data set by entering the following RACF command:

```
RACDCERT CERTAUTH EXPORT(LABEL('IBM ITSO PSIE PKI1'))  
DSN('PKI.CAPKI1.DERBIN') FORMAT(CERTDER)
```

7. To make your new certificate available to your clients, set up the /var/pkiserv/ directory by performing step 2 through step 4 in “Steps for setting up the /var/pkiserv directory” from *z/OS Security Server PKI Services Guide and Reference*, SA22-7693.

2.5.5 Renewing your PKI Services CA certificate

Eventually, your PKI Services CA certificate will expire. To avoid complications related to its expiration, you should renew the certificate before it actually expires.

Note: You will receive MVS console message IKYP026E as the expiration date approaches.

Perform the following steps to renew your PKI Services CA certificate:

1. Create a new certificate request from your self-signed CA certificate by entering the following RACF command:

```
RACDCERT CERTAUTH GENREQ(LABEL('IBM ITSO PSIE PKI1')) +  
DSN('PKI.CAPKI1.BACKUP.P12BIN')
```

2. If your PKI Services certificate authority is a root CA (that is, it has a self-signed certificate, which is the default), then generate the self-signed renewal certificate by entering the following RACF command:

```
RACDCERT CERTAUTH GENCERT('PKI.CAPKI1.BACKUP.P12BIN') SIGNWITH(CERTAUTH  
+ LABEL('IBM ITSO PSIE PKI1'))
```

3. Alternately, if your PKI Services certificate authority is an intermediate certificate authority, repeat step 2 to step 4 in “Steps for setting up the /var/pkiserv directory” from *z/OS Security Server PKI Services Guide and Reference*, SA22-7693.

2.5.6 Recovering a CA certificate profile

If the CA certificate profile in RACF is accidentally deleted, you can recover it from the backup data set.

Perform the following steps to recover a CA certificate profile:

1. Enter the following RACF commands:

```
RACDCERT CERTAUTH ADD('PKI.CAPKI1.BACKUP.P12BIN') PASSWORD(your-passphrase)  
WITHLABEL('IBM ITSO PSIE PKI1') ICSF  
RACDCERT CERTAUTH ADD('PKI.CAPKI1.DERBIN')  
RACDCERT ID(PKISTU) CONNECT(CERTAUTH LABEL('IBM ITSO PSIE PKI1')  
RING(CARING) USAGE(PERSONAL) DEFAULT)
```

Note: Password is case sensitive and must be in quotation marks. This message is issued:

IRRD119I Certificate Authority not defined to RACF. Certificate added with TRUST status. If you are not using ICSF, omit this keyword.

2. Perform the following steps to update the RACF profile with the serial number of the last CA certificate PKI Services issued. (You must restore the certificate serial number incrementer value that is stored in the profile because otherwise PKI Services resumes issuing certificates starting from serial number 1.)

- a. Ensure that PKI Services is stopped.
- b. Enter the following command from the UNIX command line to change your directory:

```
cd /usr/lpp/pkiserv/lib
```

3. Now run the iclview utility:

```
/usr/lpp/pkiserv/bin/iclview \'pki.webpki1.icl\'
```

The output from the execution of the iclview is shown in Example 2-31 on page 88. For more information about iclview, see *z/OS Security Server PKI Services Guide and Reference*, SA22-7693.

4. Record the serial number displayed (in hex) of the CA certificate listed under Cert 1. The serial number (in hex) of our CA certificate was 0.

- a. To determine your CA certificate's profile name, issue the following command to perform an unsuccessful add:

```
RACDCERT CERTAUTH ADD('PKI.CAPKI1.DERBIN') WITHLABEL('*** Bad Label ***')
```

The unsuccessful add displays this error message:

```
IRRD109I The certificate cannot be added. Profile  
00.OU=ITS0¢PSIE¢CA.0=IBM.C=US is already defined.
```

- b. Record the profile name:

Our profile name was: 00.OU=ITS0¢PSIE¢CA.0=IBM.C=US

5. Create the ICHEINTY ALTER job (Example 2-33 on page 93) in your JCL data set, replacing the highlighted values based on the information you recorded in the previous steps.

Example 2-33 Sample JCL data set for restoring the certificate serial number

```
//ANTOFFZ JOB MSGCLASS=H,CLASS=A,NOTIFY=ANTOFF
//*****
//ASM EXEC ASMACL,
// PARM.L='MAP,LET,LIST,NCAL,AC(1)'
//C.SYSIN DD *
SAMPICHE CSECT
SAMPICHE AMODE 31
SAMPICHE RMODE ANY
        STM R14,R12,12(R13) SAVE REGISTERS
        BALR R12,0
        USING *,R12 R12      = BASE REGISTER
        B                DOIT
*****
* UPDATE THE FOLLOWING DECLARES WITH YOUR CERTIFICATE INFO *
*****
ENTRY EQU *                YOUR CA CERTIFICATE PROFILE
ENTBLEN DC H'29'            LENGTH OF CERT PROFILE NAME
ENTALEN DC H'29'            LENGTH OF CERT PROFILE NAME
        DC CL18'00.OU=ITS06PSIE6CA'
        DC CL11'.0=IBM.C=US'
LSER EQU *                  CERTLSER IS AN 8 BYTE FIELD
LSERHIGH DC X'00000000'     HIGH WORD - SET TO ZERO
LSERLOW  DC X'00000000'     SET TO YOUR LAST SERIAL # (HEX)
*****
* ESTABLISH STANDARD LINKAGE *
*****
DOIT    ST R13,SAVE+4        SAVE CALLER S SAVE AREA ADDRESS
        LA R15,SAVE          GET THE NEXT SAVE AREA ADDRESS
        ST R15,8(R13)        LINK THE SAVE AREAS
        LR R13,R15           R13 POINTS TO NEXT SAVE AREA
        ICHEINTY ALTER,TYPE='GEN',ENTRYX=ENTRY,RELEASE=1.9,      X
                                SEGMENT='CERTDATA',CLASS=DIGTCLAS,ACTIONS=(A_LSER)
CLOSEUP EQU *
        L R13,SAVE+4          GET CALLER S SAVE AREA ADDRESS
        ST R15,16(R13)        SAVE ICHEINTY RC
        LM R14,R12,12(R13)    RESTORE REGISTERS EXCEPT R13
        BR R14                BACK TO INVOKER
*****
* CONSTANTS, SAVE AREAS, ETC *
*****
SAVE    DS 18F
DIGTCLAS DC CL8'DIGTCERT'
        ORG
A_LSER  ICHEACTN FIELD=CERTLSER,FLDATA=(8,LSER),RELEASE=1.9,MF=L
*****
* GENERAL EQUATES *
*****
```

```

R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
      END SAMPICHE
//C.SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
//          DD DSN=SYS1.MODGEN,DISP=SHR
//C.SYSPRINT DD SYSOUT=*
//L.SYSLMOD DD DSN=SYS1.LINKLIB(SAMPICHE),DISP=SHR
//L.SYSPRINT DD SYSOUT=*
//L.SYSIN DD *
NAME SAMPICHE(R)
/*
//RUNIT EXEC PGM=SAMPICHE
//*

```

2.5.7 Controlling applications that call R_PKIServ

Authorized applications, such as servers, that invoke the R_PKIServ callable service (IRRSPX00) can request the generation, retrieval, and administration of PKIX-compliant X.509 Version 3 certificates and certificate requests.

Applications can request end-user functions or administrative functions related to these requests. See *z/OS Security Server RACF Callable Services, SA22-7691*, for details of invoking IRRSPX00. Authorize these applications by administering RACF profiles in the FACILITY class, based on whether the application requests end user functions or administrative functions.

R_PKIServ end-user functions

The end-user functions are:

EXPORT	Retrieves (exports) a previously requested certificate.
GENCERT	Generates an auto-approved certificate.
GENRENEW	Generates an auto-approved renewal certificate.

Note: The submitted request is automatically approved.

REQCERT	Requests a certificate that an administrator must approve before it is created.
REQRENEW	Requests certificate renewal. The administrator must approve the request before the certificate is renewed.
REVOKE	Revokes a certificate that was issued previously.
VERIFY	Confirms that a given user certificate was issued by this CA and, if so, returns the certificate fields.

For end-user functions, FACILITY class profiles protect this interface. The form of the FACILITY class profiles is:

`IRR.RPKISERV.function`

Here, *function* is one of the following end-user function names in the preceding list. The user ID for the application (userid from the ACEE associated with the address space) is used to determine access:

NONE	Access is denied.
READ	Access is permitted based on subsequent access checks against the caller's user ID. To determine the caller, the current TCB is checked for an ACEE. If one is found, the authority of that user is checked. If there is no ACEE associated with the current TCB, the ACEE associated with the address space is used to locate the user ID.
UPDATE	Access is permitted based on subsequent access checks against the applicant's user ID.
CONTROL (or user ID is RACF SPECIAL)	Access is permitted, and no subsequent access checks are made.

For SAF GENCERT and EXPORT requests where the application has READ and UPDATE access, subsequent access checks are performed against the `IRR.DIGTCERT.function` FACILITY profiles. These are identical to the checks the RACDCERT TSO command makes. See *z/OS Security Server RACF Command Language Reference*, SA22-7867, for more information.

For PKI Services EXPORT, GENCERT, GENRENEW, REQCERT, REQRENEW, REVOKE, and VERIFY requests in which the application has READ and UPDATE access, subsequent access checks are performed against the

IRR.DIGTCERT.*function* FACILITY profiles. Table 2-3 summarizes the access requirements for the user ID whose access is checked.

Table 2-3 Summary of accesses required for PKI Services request

Request	Access
EXPORT	IRR.DIGTCERT.EXPORT -- UPDATE access if no pass phrase is specified on the call -- READ access if a pass phrase is specified
GENCERT	IRR.DIGTCERT.GENCERT --- CONTROL access IRR.DIGTCERT.ADD – UPDATE access if any HostIdMappings information is specified in the certificate request parameter list or the UserId field in the certificate request parameter list indicates that the certificate is being requested for a user other than the caller – READ access otherwise
GENRENEW	IRR.DIGTCERT.GENRENEW -- READ access IRR.DIGTCERT.GENCERT -- CONTROL Access Note: It is assumed that the calling application has already verified the input certificate using the VERIFY function.
REQCERT	IRR.DIGTCERT.REQCERT -- READ access
REQRENEW	IRR.DIGTCERT.REQRENEW -- READ access Note: It is assumed that the calling application has already verified the input certificate using the VERIFY function.
REVOKE	IRR.DIGTCERT.REVOKE -- READ access Note: It is assumed that the calling application has already verified the target certificate using the VERIFY function.
VERIFY	IRR.DIGTCERT.VERIFY -- READ access Note: It is assumed that the calling application has already verified that the end user possesses the private key that correlates to the input certificate.

R_PKIServ administrative functions

The administrative functions are:

CERTDETAILS	Get detailed information about one PKI Services issued certificate.
MODIFYCERTS	Change PKI Services issued certificates.
MODIFYREQS	Change PKI Services certificate requests.
QUERYCERTS	Query PKI Services issued certificates.

QUERYREQS	Query PKI Services about certificate requests.
REQDETAILS	Get detail information about one PKI Services certificate request.

For the administrative functions, a single FACILITY class profile—IRR.RPKISERV.PKIADMIN—protects this interface:

- ▶ If the caller is RACF SPECIAL, no further access is necessary.
- ▶ Otherwise, the caller needs:
 - READ access to perform read operations (QUERYREQS, QUERYCERTS, REQDETAILS, and CERTDETAILS)
 - UPDATE access for the action operations, (MODIFYREQS and MODIFYCERTS)

To determine the appropriate access level of the caller, the current TCB is checked for an ACEE. If one is found, the authority of that user is checked. If there is no ACEE associated with the current TCB, the ACEE associated with the address space is used to locate the user ID.

Attention: UPDATE access to the IRR.RPKISERV.PKIADMIN resource also controls who can act as PKI Services administrators.

Recommendation: Give UPDATE authority only to those individuals whom you would trust with the RACF SPECIAL attribute. If you do assign PKI Services administrators who do not have the RACF SPECIAL attribute, do not also give these individuals direct access to the end-user functions of the R_PKIServ callable service as described in the previous section.

2.5.8 Using encrypted passwords for LDAP servers

PKI Services uses an LDAP directory to store certificates. LDAP requires authenticating (binding) to the directory. You can do this by using a distinguished name and passwords. Passwords for binding (to multiple LDAP directories) can be encrypted or in unencrypted text. Your security policy should stipulate whether to use encrypted LDAP bind passwords. You store information about passwords in the PKI Services configuration file, pkiserv.conf. If you do not need the bind password for the LDAP server to be encrypted, you specify the values for Server1, AuthName1 and AuthPwd1 in the pkiserv.conf configuration file, otherwise you should comment them out.

Perform the following steps to use encrypted LDAP bind passwords:

1. Define a fixed-name profile LDAP.BINDPW.KEY in RACF class KEYSMSTR by entering the following command, replacing the highlighted value with your own key selected randomly using hexadecimal values from 0 to F:

```
RDEFINE KEYSMSTR LDAP.BINDPW.KEY SSIGNON(KEYENCRYPTED(A7D8E09ACDEF35AC))
```

The SSIGNON segment has two possible keywords:

KEYMASKED Used when ICSF is not active

KEYENCRYPTED Used when ICSF is active

A7D8E09ACDEF35AC is the value of the key. (Replace this with your own key.)

2. Activate the KEYSMSTR class by entering the following command:

```
SETOPTS CLASSACT(KEYSMSTR)
```

Now, you have two alternatives:

- ▶ Use profiles in RACF class LDAPBIND for each LDAP directory.
- ▶ Use fixed-name profile IRR.PROXY.DEFAULTS in class FACILITY instead of LDAPBIND class.

Profiles in class LDAPBIND

For each LDAP directory, create your own profile (in our case LDAPKI) in RACF class LDAPBIND by entering the following command:

```
RDEFINE LDAPBIND LDAPKI +  
PROXY(LDAPHOST(ldap://wtsc64.itso.ibm.com:3389) +  
BINDDN('CN=LDAPADMIN') +  
BINDPW(LDAPADMIN))
```

Note: The bind password is case sensitive.

Now you have to update the pkiserv.conf file as follows:

```
# Server1=wtsc64.itso.ibm.com:3389  
# AuthName1=CN=LDAPADMIN  
# AuthPwd1=LDAPADMIN  
-----  
BindProfile1=LDAPKI
```

Note: The first three statements are commented out. The last statement points to the profile in class LDAPBIND.

Profile in Class FACILITY

If you intend to use profile IRR.PROXY.DEFAULTS in class FACILITY instead of the LDAPBIND class for encrypted LDAP bind passwords, enter the following command to create the profile:

```
RDEFINE FACILITY IRR.PROXY.DEFAULTS +  
  PROXY(LDAPHOST(ldap://wtsc64.itso.ibm.com:3389) +  
  BINDDN('CN=LDAPADMIN') +  
  BINDPW(LDAPADMIN))
```

You have to update pkiserv.conf by commenting out all four statements:

```
# Server1=wtsc64.itso.ibm.com:3389  
# AuthName1=CN=LDAPADMIN  
# AuthPwd1=LDAPADMIN  
-----  
#BindProfile1=LDAPKI
```

As you can see, using LDAPBIND is more flexible because it enables the creation of as many profiles for LDAP servers as you would need to authenticate to. (Using FACILITY provides only one LDAP server.)

You may wish to list the segment PROXY to check your work with the RLIST command. If you are using the LDAPBIND class, enter the following:

```
RLIST LDAPBIND LDAPKI PROXY NORACF
```

This command displays the information in Example 2-34.

Example 2-34 List of the segment PROXY

CLASS	NAME
-----	----
LDAPBIND	LDAPKI

PROXY INFORMATION

LDAPHOST= LDAP://WTSC64.ITSO.IBM.COM:3389
BINDDN= CN=LDAPADMIN
BINDPW= YES

If you are using the IRR.PROXY.DEFAULTS profile of the FACILITY class, enter the following command:

```
RLIST FACILITY IRR.PROXY.DEFAULTS PROXY NORACF
```

This command displays the information in Example 2-35 on page 100.

Example 2-35 RLIST of IRR.PROXY.DEFAULTS

CLASS	NAME
----	----
FACILITY	IRR.PROXY.DEFAULTS

PROXY INFORMATION

LDAPHOST= LDAP://WTSC64.ITS0.IBM.COM:3389
BINDDN= CN=LDAPADMIN
BINDPW= YES

2.5.9 Register a Personal Certificate with RACF

This function is documented in member RACINSTL of SYS1.SAMPLIB under name SELFREG number 33. The application consists of three HTML pages and a REXX routine.

To be able to use this application, you must:

- ▶ Have one or more Personal Certificates installed in your browser.
- ▶ Have a valid user ID in RACF.

This application enables you to register a Personal Certificate with RACF, which means that you associate the certificate with your RACF user ID.

You can also de-register a previously registered Personal Certificate. When you do this, the information about your certificate is removed from RACF.

To install this application into your two Web servers' httpd.conf file:

1. Add into /web/pki1/pub/index.html and /web/pki1a/index.html reference to the application:

This is the RACF Selfreg Appl

2. Create two new directories in /web/pki1 and /web/pki1a (ocgi and rar):

mkdir /web/pki1/ocgi and mkdir /web/pki1a/ocgi
mkdir /web/pki1/rar and mkdir /web/pki1a/rar

3. Copy the three HTML pages from SYS1.SAMPLIB(RACFINSTL)—selfreg.html, selfrgft.html, and selfrghd—as files of directory rar. Copy selfreg.rexx into directory ocgi.
4. Add the following into /web/pki1/httpd.conf and /web/pki1a/httpd.conf as shown in Example 2-36 on page 101.

Example 2-36 Adding Selfreg application protection statements on both Web servers

```
Protection RAR_WebAp {  
    ServerId      RACF_AutoRegistration  
    AuthType      Basic  
    PasswdFile    %%SAF%%  
    UserID        %%CERTIF%%  
    SSL_ClientAuth client  
    Mask          All  
}  
Protect /ocgi/* RAR_WebAp  
Protect /rar/* RAR_WebAp  
-----
```

5. Add the following into the /web/pki1/httpd.conf file as shown in Example 2-37.

Example 2-37 Adding Selfreg application redirect statements on Web server webpki1

```
Redirect /PKIServ/clientauth-cgi/* https://wtsc64oe.itso.ibm.com:8013/ +  
PKIServ/clientauth-cgi/*
```

Note: Remember that you cannot use extensions (such as the plus sign above) in statements in the httpd.conf file. Use only one line to fit the statement.

6. Add the following into the /web/pki1a/httpd.conf file, as shown in Example 2-38.

Example 2-38 Adding Selfreg application pass and exec statements on Web server webpki1a

```
Exec    /ocgi/*      /web/pki1a/ocgi/*  
Pass    /rar/*      /web/pki1a/rar/*
```

Now start your browser and find your Web server address by typing

`http://wtsc64oe.itso.ibm.com:8010`

This produces the window shown in Figure 2-3 on page 102.

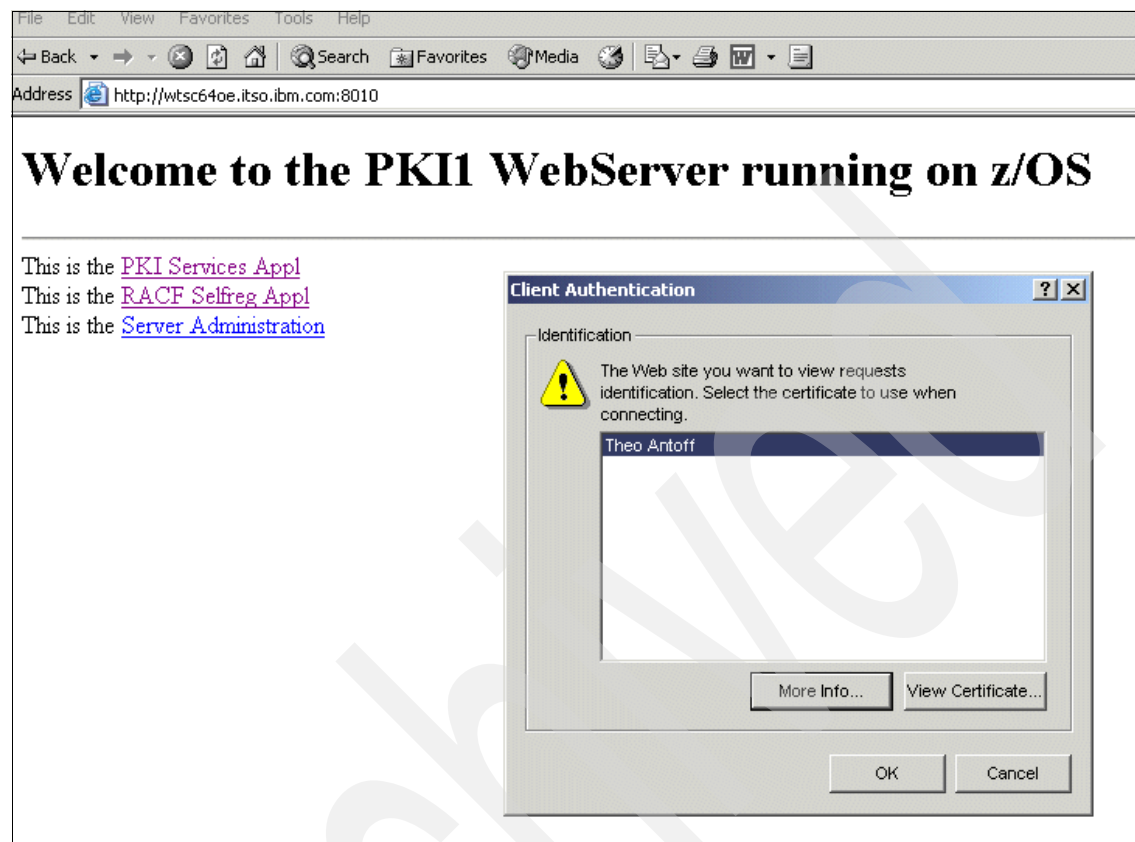


Figure 2-3 Home page for PKI Services and a prompt for certificate after selecting RACF Selfreg Appl.

The application requests a valid certificate to use it. Clicking **This is the RACF Selfreg Appl** produces the Selfreg application page shown in Figure 2-4 on page 103.

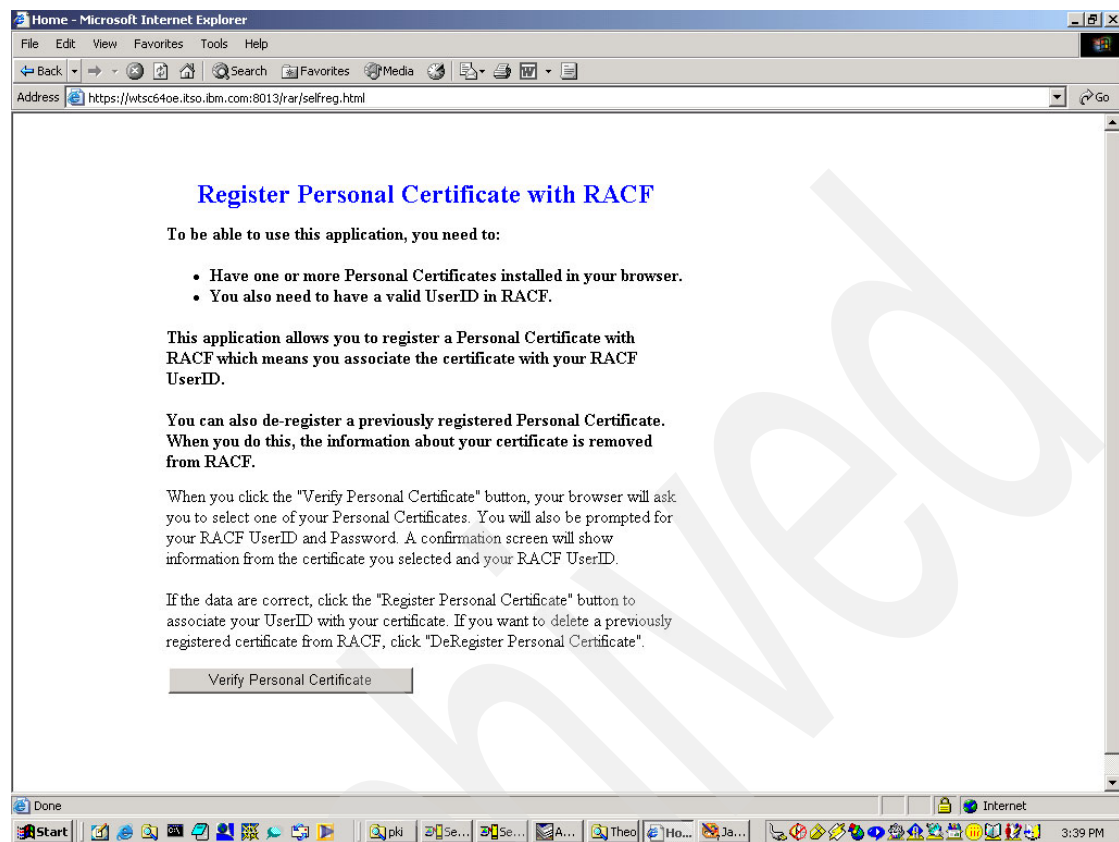


Figure 2-4 Selfreg application page

When you click the **Verify Personal Certificate** button, you are presented with a list of the personal certificates installed into your browser. After you have made a selection, you will be prompted for your RACF user ID and password and after entering them, a confirmation screen shows information from the certificate you selected and your RACF user ID. See Figure 2-5 on page 104 for details.

Verify your Certificate Information

Your RACF UserID:	ANTOFF
Certificate Common Name:	Theo Antoff
Country:	
State or Province:	
Locality:	
Organization:	IBM
Organizational Unit:	Class 1 Internet Certificate CA
Issuer Common Name:	
Issuer Country:	US
Issuer State or Province:	
Issuer Locality:	
Issuer Organization:	IBM
Issuer Organizational Unit:	ITSO PSIE CA
<input type="button" value="Register Personal Certificate"/> <input type="button" value="Deregister Personal Certificate"/>	

Figure 2-5 Confirmation certificate information

If the data is correct, click **Register Personal Certificate** to associate your user ID with your certificate. If you want to delete a previously registered certificate from RACF, go back, select the certificate from the list kept in your browser, and click **Deregister Personal Certificate**.

Now you may wish to use RACDCERT LIST to see your new certificate in RACF. It will appear last in the list of your certificates.



Easy steps to get PKI up and running

This chapter discusses how to set up PKI Services. You might modify this setup later to make it more secure or to change the appearance, but we recommend that you start easy and get it working first.

3.1 Preparing the PKI Server installation

Security Services PKI Server has several requirements for running on the system:

- ▶ Two Web servers, the PKI server task, and one or more LDAP servers.
- ▶ The first Web server runs on port 80 and 443.
- ▶ The second Web server runs on port 1433.

In a z/OS environment, you would most likely run the PKI Web servers already available in your production LPAR. Because this LPAR could already host existing Web servers on port 80 and 443, we recommend setting up TCP/IP environment additional domain names for PKI Services. This enables the use of a hidden proxy server configuration on the default ports to forward to the appropriate servers.

3.1.1 Steps to set up the PKI server

Take the following steps to set up your PKI server:

1. Create and adjust the appropriate security environment (RACF).
See Chapter 2, “RACF for PKI Services” on page 37.
2. Prepare and configure the UNIX environment.
See 3.2, “Prepare and configure the environment” on page 106.
3. Set up the Web servers.
See 3.3, “Setting up the Web servers for PKI” on page 107.
4. Configure the LDAP server.
See 3.4, “Setting up the LDAP server for PKI” on page 116.
5. Set up the PKI server task.
See 3.5, “Setting up the PKI Services task” on page 126.
6. Configure OCSF and OCEP to work with PKI Services.
See 3.6, “Configure OCSF and OCEP to work with PKI Services” on page 127.
7. Configure the PKI Services.
See 3.7, “Configure the PKI Services” on page 128.

3.2 Prepare and configure the environment

As you prepare the environment, decide where to place your configuration files for the location of additional run-time and configuration file directories for backup

or failover reasons. At the end of this chapter, we refer to a worksheet that helps you to collect and document all this informations.

Note: In a single z/OS environment there can only be one active PKI server.

IBM z/OS Security Server PKI Services Guide and Reference, SA22-7693 defines the following default locations for directories:

- ▶ PKI Services installation directory /usr/lpp/pkiserv
- ▶ PKI Services run-time directory /etc/pkiserv

In our setup, we used the corresponding Web server run-time directory /web/server1 to host the PKI server configuration and run-time files.

- ▶ PKI Services variable directory /var/pkiserv

This is the place where you store your CA certificate for public download.

Other products' default directories are:

- ▶ Web server run-time directory /etc

As indicated in several other publications, you should define a schema that supports a parallel run-time for many Web servers. We recommend using /web/server1, web/server2, and more.

- ▶ OCSF and OCEP variables directory /var/ocsf
- ▶ LDAP server run-time /etc/ldap

You may set up several LDAP servers, so you might consider placing these config files into a certain structure similar to the Web server config files.

This can be /ldap/ldap1, /ldap/ldap2, and so on.

Standard LDAP configuration is looking for a slapd.conf file or ldap.slapped.profile in this directory.

- ▶ PKI object store (OST) PKISRVD.VSAM.OST
- ▶ PKI issued certificate list (ICL) PKISRVD.VSAM.ICL

3.3 Setting up the Web servers for PKI

In this chapter we make the assumption that you have already set up the IBM HTTP Server 5.3 for z/OS (Web server) to run as a standard Web server on port 80. The setup of this Web server can be cloned to a second one because PKI Services must run at least two Web servers.

We used the following names for our Web server run-time library in HFS:
/web/pki1 and /web/pki1a.

3.3.1 Why do we need two Web servers?

The first Web server is running normal mode (port 80) and standard secure mode (port 443). It is used to request, administer, and obtain certificates.

The second Web server is running SSL mode only (normalmode off) on a special port (port 1443) and is used for client authentication. It is set up to authenticate client certificates by using an X500 server (LDAP server). This Web server is used for renew and revoke certificate actions.

3.3.2 Setting up the Web server as a secure Web server

To enable the Web server for SSL, use a server certificate. This certificate could be obtained from Certification Authorities (CAs) such as VeriSign. You can also generate a self-signed certificate.

In “Create the Web server SSL certificate” on page 70, we describe how to generate a root (CA) certificate and a server certificate signed by the CA certificate. If you already have a valid server certificate, you can jump to 3.3.3, “Customizing the Web server for SSL” on page 108.

There are several ways to generate these certificates. On z/OS you could use either the gskkyman utility or RACF services. Here we describe the RACF services method of generating a self-signed certificate.

3.3.3 Customizing the Web server for SSL

If you have already created a self-signed certificate using RACF as described in “Create the Web server SSL certificate” on page 70, then you already have this certificate in a key ring. If you obtained a certificate from an external source, ensure that it is received into a key ring.

Refer to “Create the Web server key ring” on page 70 for an explanation of how to do this.

In *IBM z/OS Security Server PKI Services Guide and Reference*, SA22-7693, the RACF key ring is named SSLring. We suggest giving this a more specific name, as there could be more key rings for different purposes. We named our key ring similar to the Web server that we used to run the PKI Services: webpki2.

To finalize your secure Web server setup, the Web server configuration file /web/pki2/httpd.conf must be customized with the configuration directives in Example 3-1.

Example 3-1 Customizing the secure Web server 1

```
# keyfile key.kdb                <- default
keyfile webpki2 SAF <- changed to RACF keyring
sslmode on                       <- default
sslport 443                     <- default
normalmode on                   <- default
```

When you have made the changes to the httpd.conf file, restart the Web server to pick up the changes.

Now you can test the secure Web server from a browser by entering:

- ▶ https://<web-server-domain-name> if you used the default SSL port 443
or
- ▶ https://<web-server-domain-name>:<SSL port number> if you are using a port number other than the default

The second Web server should be set up serving SSL only on port 1443 (or a different port such as 8443).

In this case, the definitions in Example 3-2 must be customized in /web/pki2/httpd.conf.

Example 3-2 Customizing an SSL-only Web server 2

```
# keyfile ley.kdb                <- default
keyfile webpki2 CLINETAUTH <- changed to RACF keyring
sslmode on                       <- default
sslport 1443                   <- changed to the another SSL port
normalmode off                 <- changed - no unencrypted mode
```

3.3.4 Customizing the first Web server for PKI

All of the customization parameters for PKI Services are defined in a sample httpd.conf that resides in /usr/lpp/pkiserv/samples. See “httpd.conf sample for PKI Web server 1” on page 266 for details.

We suggest that you split the file in several pieces and attach them in certain locations to your existing httpd.conf. The following sequence illustrates how you should update your httpd.conf.

In this sequence, we assume the following:

- ▶ Web server runs on ports 80 and 443.
- ▶ The other Web server runs on port 1443.
- ▶ The domain name for our PKI servers is pki.itso.ibm.com.
- ▶ SSL setup is finished.
- ▶ PKI surrogate user ID is PKISRV.
- ▶ Web server config files are in /web/server1.
- ▶ The location of the CA certificate is /var/pkiserv.

For performance reasons, we changed the location of the redirect statements in httpd.conf so that they appear before the protection statements.

Note: Add or modify the highlighted (bold) lines of the examples in your configuration files.

Note: The Web server configuration files are interpreted by the Web server from top to bottom. So it is important to customize it at the right place.

1. Search for the following lines in httpd.conf:

```
# ===== #
#
#      User authentication and document protection
#
# ===== #
```

2. Scroll a little farther, then add the Protection statements just as shown here:

```
#      The following rules allow anyone who knows your WEBADM
#      password to use the Web Server remote configuration application.
#
Protection IMW_Admin {
    ServerId      IMWEBSRV_Administration
    AuthType      Basic
    PasswdFile    %%SAF%%
    Mask          WEBADM,webadm
}

Protect /admin-bin/* IMW_Admin WEBADM
Protect /Docs/admin-bin/* IMW_Admin WEBADM
Protect /reports/* IMW_Admin WEBADM
Protect /Usage* IMW_Admin WEBADM
```

3. Add the following statements to your httpd.conf. (Some of the following text is shown in a reduced size in order to fit in on one line.)

```
#-----
# Redirection directives for the PKI server
#-----

# make sure all requests come in through SSL
Redirect /PKIServ/ssl-cgi/*      https://pki.itso.ibm.com:443/PKIServ/ssl-cgi-bin/*
#
# redirect client auth requests to the other Web server on port 1443
Redirect /PKIServ/clientauth-cgi/*  https://pki.itso.ibm.com:1443/PKIServ/clientauth-cgi/*

#-----
# End of Redirection directives for the PKI server
#-----
# Protection directives for the PKI server
#-----

Protection PublicUser {
    ServerId      PublicUser
    UserID        PKISRV
    Mask          Anyone
}
Protect /PKIServ/public-cgi/* PublicUser
Protect /PKIServ/ssl-cgi-bin/* PublicUser
Protect /PKIServ/* PublicUser

Protection AuthenticatedUser {
    ServerId      AuthenticatedUser
    AuthType      Basic
    PasswdFile    %%SAF%%
    UserID        %%CLIENT%%
    Mask          All
}
Protect /PKIServ/ssl-cgi-bin/auth/* AuthenticatedUser

Protection SurrogateUser {
    ServerId      SurrogateUser
    AuthType      Basic
    PasswdFile    %%SAF%%
    UserID        PKISRV # <-check for your actual PKI Surrogate user ID
    Mask          All
}

Protect /PKIServ/ssl-cgi-bin/surrogateauth/* SurrogateUser
#-----
# End of protection directives for the PKI server
#-----
```

4. Now search for the following lines in httpd.conf:

```
# ===== #
#
#      Mapping rules
#
# ===== #
```

5. Scroll a little farther, then add the mapping rules as illustrated:

```
# *** ADD NEW PASS RULES HERE ***

#-----
# Mapping rules for the PKI server
#-----

Exec      /PKIServ/public-cgi/*      /usr/lpp/pkiserv/PKIServ/public-cgi/*
Exec      /PKIServ/ssl-cgi-bin/*     /usr/lpp/pkiserv/PKIServ/ssl-cgi-bin/*
Pass      /PKIServ/cacerts/*         /var/pkiserv/*
#
#-----
# End of mapping rules for the PKI server
#-----
# Note that we have set up the public Web pages to reside in
/web/server1/pub
#
Pass      /*                          /web/server1/pub/*
```

6. The rest of the configuration directives (Example 3-3 on page 113) mentioned in /usr/lpp/pkiserv/samples/httpd.conf are already configured in the default setup for the Web server. You may want to check them.

```
AddType .cer application/x-x509-user-cert      ebcdic 0.5 # Browser Certificate
AddType .der application/x-x509-ca-cert         binary 1.0 # CA Certificate
```

7. Now add the additional variables to the httpd.envvars file as indicated in the sample httpd.envvars file. See Example 3-3 on page 113.

We suggest adding the following two variables to the end of the httpd.envvars:

```
_PKISERV_CONFIG_PATH=/etc/pkiserv
#_PKISERV_EXIT=/<full-path-to-pkiexit>/pkiexit
```

Here, _PKISERV_CONFIG_PATH should be the Web server run-time directory (/web/pki2) where the PKI server configuration files reside.

Note: Add the lines that are highlighted (bold) to your httpd.envvars. Some lines are split for display only.

Example 3-3 *httpd.envvars* definition for PKI Services

```
PATH=/bin:./usr/sbin:/usr/lpp/internet/bin:/usr/lpp/internet/sbin:/usr/lpp/ldap/bin:/usr/lpp/java/IBM/J1.3/bin
SHELL=/bin/sh
TZ=EST5EDT
LANG=C
LC_ALL=en_US.IBM-1047
NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lpp/internet/%L/%N:/usr/lpp/ldap/lib/nls/msg/%L/%N
LIBPATH=/usr/lpp/internet/bin:/usr/lpp/internet/sbin:/usr/lpp/ldap/lib:/usr/lpp/java/IBM/J1.3/bin
STEPLIB=CURRENT
_BPX_SHAREAS=NO
JAVA_HOME=/usr/lpp/java/IBM/J1.3

# PKI services environment variables

_PKISERV_CONFIG_PATH=/web/pki2          #<- your runtime path like /web/server1
#_PKISERV_EXIT=<full-path-to-pkiexit>/pkiexit
```

3.3.5 Customizing the second Web server for PKI

This is the setup for the second Web server that runs in SSL mode only on port 1443. It is set up to do client authentication and check the CRL against LDAP (SSLClientAuth strong).

All the customization parameters for PKI Services are defined in a sample `httpd.conf` that resides in `/usr/lpp/pkiserv/samples`. See “`httpd.conf` sample for PKI Web server 2” on page 267 for details.

We suggest that you to split the file into several pieces and add them in certain locations to your existing `httpd.conf`. The following text illustrates how you should update your `httpd.conf`.

In this example, we assume the following:

- ▶ Web server runs on port 1443.
- ▶ The other Web server runs on ports 80 and 443.
- ▶ The domain name for our PKI servers is `pki.itso.ibm.com`.
- ▶ SSL setup is finished.
- ▶ PKI surrogate user ID is `PKISRV`.
- ▶ Web server config files are in `/web/server2`.

Note: Add or modify the highlighted (bold) lines of the examples in your configuration files.

Note: The Web server configuration files are interpreted by the Web server from top to bottom. So it is important to customize it at the right place.

1. Search for the following lines in httpd.conf:

```
# ===== #
#
#       User authentication and document protection
#
# ===== #
```

2. Scroll a little farther, and add the Protection statements as shown here:

```
#       The following rules will allow anyone that knows your WEBADM
#       password to use the Web Server remote configuration application.
#
Protection IMW_Admin {
    ServerId      IMWEBSRV_Administration
    AuthType      Basic
    PasswdFile    %%SAF%%
    Mask          WEBADM,webadm
}

Protect /admin-bin/* IMW_Admin WEBADM
Protect /Docs/admin-bin/* IMW_Admin WEBADM
Protect /reports/*    IMW_Admin WEBADM
Protect /Usage*       IMW_Admin WEBADM
#-----
# Redirect directives for the PKI server
#-----
#
# redirects the public requests to the non SSL Web server
Redirect /PKIServ/public-cgi/* http://pki.itso.ibm.com:80/PKIServ/public-cgi/*
# redirects the non client auth SSL requests to the Web server on port 443 *
Redirect /PKIServ/ssl-cgi/* https://pki.itso.ibm.com:443/PKIServ/ssl-cgi-bin/*
#-----
# End of Redirect directives for the PKI server
#-----
#-----
# Protection directives for the PKI server
#-----

Protection RenewRevokeUser {
    ServerId      RenewRevokeUser
    AuthType      Basic
    UserID        PKISRV
    SSL_CLIENTAUTH Client
    Mask          Anyone
}
```



```
Protect /PKIServ/clientauth-cgi/* RenewRevokeUser
```

```
Protection AuthenticatedAdmin {
    ServerId      AuthenticatedAdmin
    AuthType      Basic

    UserID        %%CERTIF%%
    SSL_CLIENTAUTH Client
    Mask          Anyone
}
Protect /PKIServ/clientauth-cgi/auth/* AuthenticatedAdmin
```

```
#-----
# End of protection directives for the PKI server
#-----
```

3. Now search for the following lines in httpd.conf:

```
# ===== #
#
# Mapping rules
#
# ===== #
```

4. Scroll a little farther, then add the mapping rules as illustrated below:

```
# *** ADD NEW PASS RULES HERE ***

#-----
# Mapping rules for the PKI server
#-----
Exec /PKIServ/clientauth-cgi/* /usr/lpp/pkiserv/PKIServ/clientauth-cgi-bin/*
#-----
# End of mapping rules for the PKI server
#-----
# Note that we have set up the public Web pages to reside in
/web/server2/pub
Pass /* /web/server2/pub/*
```

5. Now search for the following lines in httpd.conf:

```
# SSLClientAuth directive:
#
```

6. Scroll a little farther, then add the SSLClientAuth statements as shown:

```
...
# SSLClientAuth off
SSLClientAuth strong
...
```

```

# SSLX500CARoots local_only
SSLX500CARoots local_and_x500

...
# SSLX500Host my.x500host.com
# SSLX500Host <ldap-server-name>
SSLX500Host wtsc63.itso.ibm.com

...
# SSLX500Port 22343
# SSLX500Port <ldap-port-number>
SSLX500Port 22343

...
# SSLX500UserID myUserId
# SSLX500Password myPassword
# SSLX500UserID <ldap-distinguished-name>
# SSLX500Password <ldap-password>
SSLX500UserID ADMIN
SSLX500Password secret

```

The httpd.envvars should be set up exactly as described in 3.3.4, “Customizing the first Web server for PKI” on page 109, but the `_PKISERV_CONFIG_PATH=` variable should be set to point to the second Web server run-time `/web/pki2` .

3.4 Setting up the LDAP server for PKI

This section looks at how to create the LDAP server that is needed for PKI.

We suggest that you set up more than one LDAP server for availability purposes. The first one can be local on the same image as PLI services, the other somewhere on the network. In the PKI configuration you can define several LDAP servers. The first one that is found active is the one that PKI Services uses to replicate the CA certificate and the CRL (Certificate Revocation List). For more about setting up a second LDAP server for availability, see 8.1, “Optional LDAP enhancements for availability” on page 240.

For the PKI Services LDAP environment, you might set up a special LDAP database in the TDBM backend store. We called ours LDAPPKI.

Other information needed from the LDAP environment for PKI Services is:

- LDAP domain name and port number:

```
wtsc63.itso.ibm.com:3389
```

- An admin user ID and password:

```
ADMINDN='cn=Admin'
ADMINPW='secret'
```

Instead of setting up domain name, user ID, and port, as hard coded and in unencrypted text as described here, you might choose to use an RACF definition instead. For details, see 2.5.8, “Using encrypted passwords for LDAP servers” on page 97.

The other consideration for LDAP for PKI Services is how you set up your LDAP tree. We decided to set up a tree based on the suffix `cn=US`, which allowed us to use just one suffix. `cn=US` is the country definition for our CA setup.

The setup for the LDAP server follows the basic installation procedures for the TDBM as described in *z/OS Security Server LDAP Server Administration and Use*, SC24-5923. Your environment influences the way you set up your LDAP server, so we offer the base requirements for the installation of the PKI LDAP server. Table 3-1 is a checklist of the required steps.

Table 3-1 LDAP checklist

Action Item	Description	Explanation
Build the RACF environment to support the LDAP server.	A one-time step that creates the LDAP server's RACF user ID and group with an OMVS segment. Defines the user/group as a UNIX daemon and server and gives the LDAP server access to the appropriate resources.	RACF commands are defined in the earlier chapter about RACF commands.
Create the LDAP profiles and run the <code>ldapcnf</code> utility.	Update the appropriate LDAP profiles for the DB2 and <code>slapd</code> environments and run the <code>ldapcnf</code> utility to build the jobs to create the PKI LDAP server.	Described in detail here and in the LDAP server manual (SC24-5923)

Action Item	Description	Explanation
Build the DB2 (TDBM) environment for your LDAP server.	Create the DB2 database, tablespaces, tables, and indexes, as well as the CLI interface. Size depends on the use of the LDAP server, but the default sizes of the DB2 database are probably good for a medium-size PKI environment with an isolated LDAP server.	Described in detail in this chapter and in the LDAP server manual (SC24-5923).
Ensure that the slapd.conf file is correctly built from the ldapcnf utility.	Review the global and TDBM parameters in the configuration file. Ensure that the TCP/IP and DB2 interfaces are correct.	The basic requirements are described below. Several enhancements and additional features are described in the PKI Infrastructure Additional Features Chapter later.
Set the LDAP server's PROC in the system proclib.	Move the LDAP PROC in the appropriate place and ensure that any changes to the JCL and parameters are correct.	Described in this section.
Start the LDAP and test the connectivity.		Described in this section.
Load the required schema definitions into the LDAP server.	The PKI environment requires that the LDAP server is able to handle certain data types and characteristics. These are defined by the schema, which must be loaded into the DB2 (TDBM) environment.	Described in this section and in the LDAP server manual (SC24-5923).
Define the PKI suffix(es) and the admin user ID, and test the results.	Define the suffix(es) that match the ones used in the PKI templates and define the LDAP administrator ID and password. List them out to ensure that they are defined correctly.	Described in this section.

Action Item	Description	Explanation
Update the configuration file and restart the LDAP server. Test your results.		Described in this section.

3.4.1 LDAP setup: running the ldapcnf utility

The LDAP files are installed into the UNIX subdirectory `/usr/lpp/ldap`. To build the LDAP server using the `ldapcnf` utility, copy the following files into a work directory from the `/usr/lpp/ldap/etc` directory:

- ▶ `ldap.profile`
- ▶ `ldap.db2.profile`
- ▶ `ldap.slapd.profile`
- ▶ `ldap.racf.profile`

As the RACF environment was built as described in the previous chapter, this will describe the DB2 and TCP/IP interfaces with the LDAP server. For a complete explanation of these files and the use of `ldapcnf`, read *z/OS Security Server LDAP Server Administration and Use*, SC24-5923. In the `ldap.profile` file, the important parameters that must be set and matched with the PKI templates and your system environment are:

- ▶ `TDBM_SUFFIX`, which is the suffix that will be created within the LDAP configuration file. It should match the suffix with the PKI templates.
- ▶ `LDAPUSRID` and `LDAPUSRGRP`, which are the LDAP server's RACF user ID and group. These should match what was defined previously.
- ▶ `ADMINDN` and `ADMINPW`, which are the LDAP server's administrator. This must match what is in the PKI templates unless you are going to access controls within LDAP. The `ADMINPW` is used for the initial installation of the LDAP server and will be changed when the LDAP administrator is defined within the LDAP directory. Either the changed password must match what is coded into the PKI templates, or you could use an RACF user ID if the SDBM is set up, or you could use the new `LDAPBIND` features.
- ▶ The rest of the parameters are `HLQ` for system data sets, locations of required information for the `ldapcnf` utility, and so on. All of these must match what is in your system environment for the `ldapcnf` utility to work correctly.
- ▶ One last important parameter is the location of the output from the `ldapcnf` utility. The output will be placed in the data set indicated by the `OUTPUT_DATASET` parameter.

In our case, the important part of the `ldap.profile` file appeared as shown in Example 3-4 on page 120.

Example 3-4 Our LDAP profile file

```
OUTPUT_DATASET='JJONES.LDAPOUT'  
LDAPUSRID='LDAPSRV'  
LDAPUSRGRP='LDAPGRP'  
TDBM_SUFFIX='c=US'  
ADMINDN='cn=Admin'  
ADMINPW='secret'
```

Within the ldap.slapped.profile file, the following parameters must be set appropriately:

- ▶ LDAP_HOSTNAME, which is the IP address of your LDAP server.
- ▶ PORT, which is the unsecure port that your LDAP server is listening on.
- ▶ There are other parameters that can be set if you are using SSL, replication, SDBM, multiserver mode, password encryption, and so on. As these are not required, they are not discussed here.

In our case, the important part of the ldap.slapped.profile appeared as in Example 3-5.

Example 3-5 LDAP slapd.profile file

```
LDAP_HOSTNAME='wtsc63.itso.ibm.com'  
PORT='3389'
```

Nothing within the ldap.db2.profile is required to match the PKI environment, but the LDAP server and DB2 must communicate correctly for the LDAP server to be useful. If you are not responsible for DB2 at your installation or you do not have DBA authority, then the easiest way to handle the ldap.db2.profile is to set the following parameters and run the rest with the defaults:

- ▶ TDBM_DB2_LOCATION, which is the DDF location name. Run with the default of LOC1.
- ▶ TDBM_DB2_USERID, which is the HLQ of your DB2 tables. Set this the same as your LDAP server RACF user ID.
- ▶ TDBM_DB2_DBNAME, which is the name of your DB2 database. Set this to the appropriate value for your installation or something that is meaningful, such as PKILDAP.

In our case, the ldap.db2.profile looked similar to Example 3-6.

Example 3-6 LDAP db2.profile file

```
TDBM_DB2_LOCATION='DB2H'  
TDBM_DB2_USERID='LDAPSRV'  
TDBM_DB2_DBNAME='LDAPPKI'
```

Then, run the `ldapcnf` utility using the command from your work directory (Example 3-7).

Example 3-7 ldapcnf utility run

```
/usr/lpp/ldap/sbin/ldapcnf -i ldap.profile
```

This creates several members in the output data set that you specified in the `ldap.profile` file.

DB2 (TDBM) for LDAP setup

To build the DB2 environment, first you must have DBA authority. If you do not have the authority to create and maintain databases, then hand over the DBCLI, DBSPUFI, and DSNAOINI members to your DBA to create the appropriate databases. In return, you need:

- ▶ The data set with the DSNAOINI file
- ▶ The DB2 data source or location name
- ▶ The user ID that own or created the DB2 tables
- ▶ The database name
- ▶ The DB2 subsystem ID

The only authority that is required within the DB2 environment is that the LDAP server's RACF user ID must have DBADM authority to the newly created database, EXECUTE authority on the CLI plan, and SELECT authority on SYSIBM.SYSCOLUMNS.

If you are the DBA for your installation, then review the DBSPUFI member of the output data set. Ensure that the buffer pool and storgroup are set the way that you want them. Also, review the sizes of the table spaces for your environment. The defaults are pretty good for a medium-size PKI environment if you are not sharing the LDAP server with any other applications. Then run the DBSPUFI member using the SPUFI application. This should get a zero return code on the COMMIT. If you have not run a CLI plan, then run the DBCLI plan and ensure that the LDAP server's RACF user ID has EXECUTE authority to the plan name. Finally, check the DSNAOINI file. It is critical that the DB1 subsystem name, location name, and CLI plan name are correct. Also indicate whether you are using CAF or RRSF in DB2.

LDAP PROC and the configuration file

When the DB2 environment is complete, review the JCL in the LDAP PROC. This is placed in the output data set under the name you specified. Review this member to ensure that the JCL is correct, and move it into the appropriate system proclib. As all of our system data sets are in LINKLST but our DB2 data sets are not, a couple of changes were made to the default PROC, and our final run-time LDAP PROC appeared as in Example 3-8 on page 122.

Example 3-8 LDAP procedure

```
//LDAPCRL PROC REGSIZE=0M,  
//*-----  
// PARMS=' ',  
// PCNFOUT='JJONES.LDAPOUT',  
// OUTCLASS='H'  
//*-----  
//GO      EXEC PGM=GLDSLDAP,REGION=&REGSIZE,TIME=1440,  
//        PARM=(' /&PARMS >DD:SLAPDOUT 2>&1' )  
//*-----  
//STEPLIB DD DSN=DB2H7.SDSNLOAD,DISP=SHR  
//CONFIG  DD DSN=&PCNFOUT.(SLAPDCNF),DISP=SHR  
//ENVVAR  DD DSN=&PCNFOUT.(SLAPDENV),DISP=SHR  
//SLAPDOUT DD SYSOUT=&OUTCLASS  
//SYSOUT  DD SYSOUT=&OUTCLASS  
//SYSUDUMP DD SYSOUT=&OUTCLASS  
//CEEDUMP DD SYSOUT=&OUTCLASS
```

The other item to check here is the configuration and the environment variables files. The environment variables file (envvars) are in the SLAPDENV member of the output data set. There is nothing to change here at this time, but as you install OCSF and ICSF you might add a LIBPATH parameter in this file. The other way to tell the LDAP server where these executables are is via the PARM parameter within the PROC.

The LDAP configuration file is in the SLAPDCNF member of the output data set. The important global parameters are:

- ▶ LISTEN, which defines the IP address and port of the LDAP server.
- ▶ ADMINDN, which defines the LDAP server's administrator. On the initial installation of the LDAP server, you also have to set up the adminPW parameter, which is the LDAP server's administrator's password. This password is only needed for the administrator as defined within the LDAP directory.

The important TDBM parameters are:

- ▶ SUFFIX, which is the beginning of this portion of the LDAP directory. There can be more than one of these.
- ▶ SERVERNAME, which is the DB2 location.
- ▶ DBUSERID, which is the name of the creator of the DB2 tables.
- ▶ DATABASENAME, which is the name of the DB2 database.
- ▶ DSNAOINI, which indicates where the CLI interface definition file is.

Our SLAPDCNF member appeared as in Example 3-9 on page 123.

Example 3-9 SLADPCNF member

```
maxConnections 200
listen ldap://wtsc63.itso.ibm.com:3389
adminDN cn=Admin
adminPW passon
# logfile //DD:LOGOUT
# -----
database tdbm GLDBTDBM
suffix "c=US"
dsnaoini JJONES.LDAPOUT(DSNAOINI)
servername DB2H
dbuserid JJONES
databasename LDAPPKI
```

Starting the LDAP server and loading the schema

With the RACF, TCP/IP, PROC, and DB2 features set up for LDAP, the server is now ready to be started. Go into SDSF and issue the following command (assuming that `ldapcr1` is the LDAP member within your proclib):

```
/s ldapcr1
```

As you watch the LDAP server come up, you should see the following message in either the SYSLOG or the JES messages for the PROC:

```
GLD0122I Slapd is ready for requests.
```

Ensure that there were now DB2 error messages in the JES messages within the PROC. Then go into OMVS and change directories to your work directory. Now issue the following command:

```
ldapsearch -h wtsc63.itso.ibm.com -p 3389 -V 3 -b "" -s base -L "objectclass=*"
```

If your LDAP server is set up and communicating correctly, then you should see some output that looks similar to Example 3-10.

Example 3-10 Output from ldapsearch command

```
dn:
supportedcontrol: 2.16.840.1.113730.3.4.2
supportedcontrol: 1.3.18.0.2.10.2
supportedcontrol: 1.3.18.0.2.10.10
supportedextension: 1.3.6.1.4.1.1466.20037
namingcontexts: c=us
subschemasubentry: CN=SCHEMA,c=us
supportedsaslm mechanisms: EXTERNAL
supportedsaslm mechanisms: CRAM-MD5
supportedsaslm mechanisms: DIGEST-MD5
supportedldapversion: 2
supportedldapversion: 3
```

At this point the schema must be loaded into the LDAP directory. First, copy schema.user.ldif and schema.IBM.ldif from the /usr/lpp/ldap/etc directory into your working directory. You really only need the schema.user.ldif file to support the PKI environment, but in case you ever want to use this LDAP server for anything else that requires a more extensive schema, you could include both files. Edit each one of these files, changing <suffix> at the front of these files to match the suffix in your configuration file. In our case, we issued this command while we were editing the files:

```
c '<suffix>' 'c=us'
```

The top non-comment line in both files should be 'cn=schema, c=us'. Save the files, then from your working directory, issue the following commands (where cn=admin and secret match up with your configuration file and /u/jjones/ldapprod is your working directory):

```
ldapmodify -h wtsc63.itso.ibm.com -p 3389 -D cn=admin -w secret \  
-f /u/jjones/ldapprod/schema.user.ldif  
ldapmodify -h wtsc63.itso.ibm.com -p 3389 -D cn=admin -w secret \  
-f /u/jjones/ldapprod/schema.IBM.ldif
```

The output from these commands is only one line, indicating that the LDAP server is being modified. Any other message means that there is some sort of problem and it should be fixed. You can view the schema that you have just loaded by issuing the following command:

```
ldapsearch -h wtsc63.itso.ibm.com -p 3389 -s base -b "cn=schema,c=us" \  
"objectclass=subschema"
```

Be prepared for several screens of output.

Defining the suffix and administrator

The final step in making the LDAP server ready for PKI Services is to define the suffix. (This is the suffix that was indicated in the PKI template.) To define the suffix and the administrator that we are using in our examples, issue the following command from your OMVS environment (where /u/jjones/ldapprod is your working directory):

```
ldapadd -h wtsc63.itso.ibm.com -p 3389 -D cn=admin -w secret \  
-f /u/jjones/ldapprod/suffix_admin.ldif
```

The LDIF data in the suffix_admin.ldif file is as shown in Example 3-11 on page 125.

Example 3-11 ldif data in suffix_admin.ldif

```
dn: c=us
objectclass: top
objectclass: country
c: us

dn: cn=admin,c=us
objectclass: top
objectclass: person
objectclass: inetorgperson
objectclass: organizationalPerson
cn: admin
sn: admin
userPassword: secret
```

This defines the suffix and the LDAP administrator, with its password, in the LDAP directory. When you issue the command to add this data to the LDAP server you will get messages, outside of the message that you are adding data, from the LDAP server if there are errors. To list the data and verify that your data has been added correctly, issue this command from the OMVS environment (where ‘\’ is the UNIX continuation symbol):

```
ldapsearch -h wtsc63.itso.ibm.com -p 3389 -D cn=admin -w secret -b c=us \
“objectclass=*”
```

This command should produce output that looks similar to Example 3-12.

Example 3-12 ldapsearch output

```
c=us
objectclass=top
objectclass=country
c=us

cn=admin,c=us
objectclass=top
objectclass=inetorgperson
objectclass=person
objectclass=organizationalPerson
cn=admin
sn=admin
userpassword=secret
```

Now we are ready to make the LDAP server production-ready. First, in SDSF, stop the LDAP server with the **/p ldapcr1** command. When the LDAP server is stopped, edit the SLAPDCNF member of your output data set to remove the

adminPW parameter, and adjust the adminDN to add the suffix to the DN. Now your SLAPDCNF member should look similar to Example 3-13.

Example 3-13 Updated slapdcnf member

```
maxConnections 200
listen ldap://wtsc63.itso.ibm.com:3389
adminDN cn=Admin,c=us
# logfile //DD:LOGOUT
# -----
database tdbm GLDBTDBM
suffix "c=US"
dsnaoini JJONES.LDAPOUT(DSNAOINI)
servername DB2H
dbuserid JJONES
databasename LDAPPKI
```

After you restart the LDAP server, it has the basic requirements to work with PKI Services.

3.5 Setting up the PKI Services task

PKI Services use VSAM data sets for object store (OST) and issued certification list (ICL). The PKISTU started task is used to manage these data bases.

To create these data sets, use the IKYCVSAM sample job in SYS1.SAMPLIB.

We changed the default VSAM data set names to fit our site standards. For this test, we decided to prefix the data sets with PKI, using WEBPKI2 as the second qualifier.

Example B-7 on page 330 shows the JCL we used to create the VSAM data sets.

When these data sets have been created, set up the PKISRV2 started task.

We recommend that you copy PKISRVD to a started task procedure name that fits your environment. In our case, we copied it to PKISRV1, PKISRV2.

Edit the appropriate PKISRVD procedure to configure the data set names (Example 3-14).

Example 3-14 SYS1.PROCLIB(PKISRV1)

```
//*****
//*
//*          Licensed Materials - Property of IBM
//*          5694-A01
//*
```

```

/*          (C) Copyright IBM Corp. 2001          *
/*          Status=HKY7706                        *
/*          *                                      *
/*          *****                              *
/*          *****                              *
/*          *                                      *
/*          Procedure for starting the PKI Services Daemon *
/*          *                                      *
/*          *****                              *
//PKISRV1  PROC REGSIZE=512M,                      X
//          OUTCLASS='*',                          X
//          TZ='EST5EDT',                          X
//          FN='pkiserv.envars',                    X
//          DIR='/web/pki2',                        X
//          STDO='1>DD:STDOUT',                     X
//          STDE='2>DD:STDERR'
/*-----
//GO      EXEC  PGM=IKYPKID,REGION=&REGSIZE,TIME=1440,
//      PARM=('ENVAR("_CEE_ENVFILE=&DIR/&FN","TZ=&TZ") / &STDO &STDE')
//STDOUT  DD  SYSOUT=&OUTCLASS
//STDERR  DD  SYSOUT=&OUTCLASS
//SYSOUT  DD  SYSOUT=&OUTCLASS
//CEEDUMP DD  SYSOUT=&OUTCLASS
/* ObjectDSN data set
//OST      DD  DSN=PKI.WEBPKI1.OST,
//      DISP=SHR,AMP=('BUFNI=6,BUFND=4')
/* ObjectTidDSN data
//TID      DD  DSN=PKI.WEBPKI1.OST.PATH,
//      DISP=SHR,AMP=('BUFNI=6,BUFND=4')
/* ObjectStatusDSN data set
/* ICLDSN data set
//ICL      DD  DSN=PKI.WEBPKI1.ICL,
//      DISP=SHR,AMP=('BUFNI=6,BUFND=4')
/* ICLStatusDSN data set
//ISTAT    DD  DSN=PKI.WEBPKI1.ICL.STATUS,
//      DISP=SHR,AMP=('BUFNI=1,BUFND=4')
/* ICLRequestorDSN data set
//IREQ     DD  DSN=PKI.WEBPKI1.ICL.REQUESTR,
//      DISP=SHR,AMP=('BUFNI=1,BUFND=4')

```

3.6 Configure OCSF and OCEP to work with PKI Services

In this chapter we assume that you already set up OCSF, so ensure that the files in /var/ocsf exist.

In order to run PKI Services with OCSF, you must set up the PKI Services Trust Policy (PKITP) plug-in for OCSF. The PKITP performs certificate validation.

There are two shell scripts in /usr/lpp/pkiserv/bin that perform the installation and verification of the PKITP setup. To install PKITP:

1. Run the PKITP installation routine:

```
su
cd /usr/lpp/pkiserv/lib
/usr/lpp/pkiserv/bin/install_pktp
```

2. It returns some questions:

```
addin directory?
/usr/lpp/pkiserv/lib
addin filename?
pktp.so
action? "install|uninstall"
install
```

3. After this is complete, you can run the verification:

```
TRAUNER:/Z04RC1/usr/lpp/pkiserv/lib: >/usr/lpp/pkiserv/bin/pktp_ivp
Starting pktp IVP
Initializing CSSM
CSSM Initialized
Attaching pktp
Attach successful, Detaching pktp
Detach of pktp successful
Completed pktp IVP
TRAUNER:/Z04RC1/usr/lpp/pkiserv/lib: >
```

3.7 Configure the PKI Services

Start copying some of the sample files provided with PKI Services from the installation directory /usr/lpp/pkiserv/samples to the run-time directory. We set this directory to be the same as the Web server Cryptographic Coprocessor /web/pki2.

Be careful because the samples directory contains some files that would overwrite your Web server configuration files if you copied all files. Therefore, we suggest:

1. In the UNIX shell, ensure that you have superuser authority (or at least authority to rename and move files).
2. Copy all of the necessary files to your Cryptographic Coprocessor:

```
cd /usr/lpp/pkiserv/samples
cp pkis*.* /web/server1
```

The httpd configuration files are unnecessary because the Web servers are already configured.

We will copy the .forms files later when needed.

Now set up the directory that hosts your CA certificate.

As discussed in 3.2, “Prepare and configure the environment” on page 106, we used the directory /var/pkiserv to host the CA certificate. For more about this, read “Save the CA certificate to a data set for import to a UNIX file” on page 68.

3.7.1 Set up the environment variables for PKI Services

Now, configure the PKI environment variables file.

Note: The PKI environment variables file is called *pki.envvars* instead of httpd.envvars, which is the environment variables file for the Web server.

Verify or change the _PKISERV_CONFIG_PATH and the OCSFREGDIR directory information (Example 3-15 on page 129).

The variable _PKISERV_MSG_LEVEL defines the debug level for PKI Services. We recommend using debug level D for all components when you install PKI Services for the first time. The default level for all components is W.

More about component levels and debug levels can be found in *IBM z/OS Security Server PKI Services Guide and Reference*, SA22-7693, and in “PKI Services subcomponents and message levels” on page 330.

You might change these values while the PKI server is running by using a modify command to the PKI server. This is described in Chapter 4, “Customizing the z/OS PKI Services: the template file” on page 137.

Example 3-15 pki.envvars customization

```
#-----#
#                                           #
# PKI Services sample environment variable file      #
#                                           #
# Licensed Materials - Property of IBM              #
# 5694-A01                                           #
# (C) Copyright IBM Corp. 2001, 2002                #
# Status = HKY7707                                  #
#                                           #
#-----#
#
# Language and Path configurations
```

```

#
LANG=En_US.IBM-1047
PATH=/usr/sbin
LIBPATH=/usr/lpp/pkiserv/lib:/usr/lib
NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lpp/pkiserv/lib/nls/msg/%L/%N
#
# Configuration File location and Message configuration Options
#
#_PKISERV_CONFIG_PATH=/etc/pkiserv
PKISERV_CONFIG_PATH=/web/server1          <- Change to the actual runtime directory
PKISERV_MSG_LOGGING=stdout_logging
#_PKISERV_MSG_LEVEL=*.w
PKISERV_MSG_LEVEL=*.d
#
# Location of the OCSF Registry (/var/ocsf is the default location)
#
OCSFREGDIR=/var/ocsf

```

3.7.2 Customizing the PKI Services configuration file

In this section, we describe the configuration directives that must be customized in order to get PKI Services up and running. More customization directives are described in Chapter 4, “Customizing the z/OS PKI Services: the template file” on page 137.

The first directives to look at in the `pkiserv.conf` file are the VSAM file names that were prepared in 3.5, “Setting up the PKI Services task” on page 126 for the OST and the ICL. The default for these data sets is `PKISRVD.VSAM.OST` and `PKISRVD.VSAM.ICL`, which we changed to `PKI.WEBPKI(n)`. If you changed these names, then you must change the configuration directives accordingly. If you defined the DD names of the VSAM data sets in the procedure, then we recommend that you refer to the DD names:

```

[ObjectStore]
# Data set name of the VSAM request (object store) base CLUSTER
# ObjectDSN='pkisrvd.vsam.ost'
ObjectDSN=DD:OST

# Data set name of the VSAM object store PATH for the transaction ID (TID)
alternate index
# ObjectTidDSN='pkisrvd.vsam.ost.path'
ObjectTidDSN=DD:TID

# Data set name of the VSAM issued certificate list (ICL) base CLUSTER
# ICLDSN='pkisrvd.vsam.icl'
ICLDSN=DD:ICL

```



```
# Data set name of the VSAM ICL PATH for the status alternate index
# ICLStatusDSN='pkisrvd.vsam.icl.status'
ICLStatusDSN=DD:ISTAT

# Data set name of the VSAM ICL PATH for the requestor alternate index
# ICLRequestorDSN='pkisrvd.vsam.icl.requestr'
ICLRequestorDSN=DD:IREQ
```

In the [SAF] section of pkiserv.conf, define the right key ring to be used. This certificate had been added to the key ring by the RACF setup using the RACDCERT ADDRING(CARING) ID(PKISTU) command. In our example, we used the label ITSOCASC63 for our CA certificate:

```
RACDCERT ADDRING(ITSOCASC63) ID(PKISTU)
```

Now this information must be defined in pkiserv.conf:

```
[SAF]
#KeyRing=PKISRVD/CARing
KeyRing=PKISTU/ITSOCASC63
```

In the [LDAP] section, define the LDAP server information such as server-domain-name, port name, admin user ID, and password. Ensure that this is consistent with the LDAP definition in 3.3.5, “Customizing the second Web server for PKI” on page 113.

```
[LDAP]
NumServers=1
PostInterval=5m
# Server1=myldapserver.mycompany.com:389
Server1=wtsc63.itso.ibm.com:3389      <- domain name and port of LDAP
AuthName1=CN=admin                  <- Authentication name
AuthPwd1=secret                     <- Password
CreateOUValue= Created by PKI Services
RetryMissingSuffix=T
# Name of the LDAPBIND Class profile containing the bind information for
LDAP server 1. This key is optional. Used in place of keys Server1,
AuthName1. and AuthPwd1
#BindProfile1=LOCALPKI.BINDINFO.LDAP1
```

For a more secure environment, we recommend using the LDAPBIND class instead of having user IDs and passwords floating around unencrypted. This is described in 2.5.8, “Using encrypted passwords for LDAP servers” on page 97.

Another parameter you might change in this section is the number of LDAP servers, NumServers=1. For availability reasons, you should refer to at least two LDAP servers. See Chapter 8, “LDAP enhancements for availability” on page 239.

If you want to post the information to LDAP in a time frame other than every five minutes, then you should change `PostInterval=5m` to another value, such as `PostInterval=3m` (3 minutes).

We can also configure the [General] section, even if we do not use its function for now:

```
[General]
InitialThreadCount=10

# full pathname or data set name containing the 'your certificate is ready'
# message form. Defaults to no message issued
# ReadyMessageForm=/etc/pkiserv/readymsg.form
ReadyMessageForm=/web/pki2/readymsg.form      <- runtime

# full pathname or data set name containing the 'your certificate request
# has been rejected' message form. Defaults to no message issued
# RejectMessageForm=/etc/pkiserv/rejectmsg.form
RejectMessageForm=/web/pki2/rejectmsg.form

# full pathname or data set name containing the 'your certificate is about
# to expire' message form. Defaults to no message issued
# ExpiringMessageForm=/etc/pkiserv/expiringmsg.form
ExpiringMessageForm=/web/pki2/expiringmsg.form
```

3.7.3 Customizing the PKI template

The PKI template file is used by the various REXX CGI programs to obtain variables and to set up the HTML output for the Web pages.

The customization of the pages is described in Chapter 4, “Customizing the z/OS PKI Services: the template file” on page 137. In this section, however, we describe the necessary configuration steps to make PKI Services operational.

PKI Services can create certificates through different ways. One way is purely through RACF and is called a SAF certificate. It is approved automatically after host user ID and password verification. This is the historic way versus the PKI certificates that can be issued and administered now using PKI Services.

To enable SAF certificates, customize the `%%SignWith=SAF:CERTAUTH/taca%%` parameters in `pkiserv.tmpl`. Change the `taca` parameter RACF label of your CA certificate. This parameter is found two times in `<CONSTANT>` sections.

```
<CONSTANT>
%%KeyUsage=handshake%%
%%NotAfter=365%%
```

```

%%SignWith=SAF:CERTAUTH/taca%%
%%SignWith=SAF:CERTAUTH/ITSO CA SC63%%
</CONSTANT>

<CONSTANT>
%%KeyUsage=handshake%%
%%NotAfter=365%%
%%OrgUnit=SAF template certificate%%
# %%OrgUnit=Nuts and Bolts Division%%
%%OrgUnit=ITSO Poughkeepsie, NY%%
# %%Org=The Firm%%
%%Org=IBM%%
%%Country=US%%
%%SignWith=SAF:CERTAUTH/taca%%
%%SignWith=SAF:CERTAUTH/ITSO CA SC63%%
%%CommonName=%%
</CONSTANT>

```

This label points to the CA in RACF that is needed to sign the certificates.

The example also shows other modifications that you should do:

- ▶ Change the Org parameter from The Firm to the name of your company or organization.
- ▶ Change the OrgUnit parameter in the same way, unless your organizational unit's name really is Nuts and Bolts Division.
- ▶ Change the Country parameter to the country you are located, or add a Country parameter if it does not exist

There is no country definition for PKI certificates in the sample template file. We recommend adding one and using this country (C=US) as the tree entry point in LDAP.

Note: Be aware that changes in these fields change the content of your certificates. This is also sensitive to LDAP.

Search pkiserv.tmpl for <CONSTANT>. There are several places to modify:

```

<CONSTANT>
%%KeyUsage=handshake%%
%%NotAfter=365%%
%%OrgUnit=SAF template certificate%%
# %%OrgUnit=Nuts and Bolts Division%%
%%OrgUnit=ITSO Poughkeepsie, NY%%
# %%Org=The Firm%%
%%Org=IBM%%
%%Country=US%%

```

```

%%SignWith=SAF:CERTAUTH/taca%%
%%SignWith=SAF:CERTAUTH/ITSO CA SC63%%
%%CommonName=%%
</CONSTANT>

<CONSTANT>
%%NotBefore=0%%
%%NotAfter=365%%
%%KeyUsage=handshake%%
%%OrgUnit=Class 1 Internet Certificate CA%%
%%OrgUnit=ITSO Poughkeepsie, Class 1 Internet Certificate CA%%
%%Org=The Firm%%
%%Org=IBM%%
%%Country=US%%
%%SignWith=PKI:%%
</CONSTANT>

```

Other customizations in pkiserv.temp can be done to add fields or to change the look. This is described in Chapter 4, “Customizing the z/OS PKI Services: the template file” on page 137.

3.8 Checking the VSAM data set

For troubleshooting purposes, two utilities browse VSAM data sets directly:

vosview	Lists the VSAM object store
iclview	Lists the VSAM-issued certificate data set

To use these utilities define the bin and lib directories in your personal .profile file:

```

export PATH=$PATH:/usr/lpp/pkiserv/bin
export LIBPATH=$LIBPATH:/usr/lpp/pkiserv/lib

```

To view the object store, use the **vosview** command `vosview \'VSAMOSTDSNAME\'`. This command returns the object key that is valuable to trace problems in the PKI Services task (for example, when a posting to LDAP is not successful). It also returns the status of the certificate.

Example 3-16 vosview VSAM listing

```

vosview \'trauner.pkisrzd.webpki2.vsam.ost\'

```

```

-----
Object key = 205
name = "newman@mad.org"
tid = 1jJwJZSErRNI2Qn06U+++++
appldata = 1YBSSL
comment = ok
data len = 1329

```

```
flags = 2022010 - Type = Cert   State = CA CertSigned   Complete
Creation time is: 2003/04/29 17:59:21
Last modified time is: 2003/04/29 18:01:21
```

```
-----
Object key = 208
name = "Roland Trauner"
tid = 1jJwMqFqiLlK2Qn06U+++++
apldata = 1YBSSL
comment =
data len = 1527
flags = 2022010 - Type = Cert   State = CA CertSigned   Complete
Creation time is: 2003/04/29 18:43:12
Last modified time is: 2003/04/29 18:45:53
-----
```

Example 3-17 shows how to view the issued certificates.

Example 3-17 iclview VSAM listing

```
iclview \'trauner.pkisrwd.webpki2.vsam.ic1\'
Cert 21: newman@mad.org
      REVOKED, on posted CRL ()
      Issued at 2003-04-29 18:01:02
      Last changed 2003-04-29 20:28:59
      Subject: MAIL=newman@mad.org,CN=Alfred E. Newman,OU=ITSO
Poughkeepsie\, Class 1 Internet Certificate CA,0=IBM,C=US
      Issuer: CN=IBM ITSO POK ROOT CA SC63,OU=IBM ITSO POUGHKEEPSIE,C=US
      Requester: newman@mad.org
      AppData: 1YBSSL
      Serial Number: 18
      Email flag: Off

Cert 22: Roland Trauner
      ISSUED (Issued certificate)
      Issued at 2003-04-29 18:44:01
      Last changed 2003-04-29 18:44:01
      Subject: MAIL=trauner@de.ibm.com,CN=Roland Trauner\, IBM T8,OU=ITSO
Poughkeepsie\, Class 1 Internet Certificate CA,0=IBM,C=US
      Issuer: CN=IBM ITSO POK ROOT CA SC63,OU=IBM ITSO POUGHKEEPSIE,C=US
      Requester: Roland Trauner
      AppData: 1YBSSL
      Serial Number: 19
      Email flag: On
-----
```


Customizing the z/OS PKI Services: the template file

In this chapter we explain how the external look and behavior of the z/OS PKI Services are dictated by the contents of the template (pkiserv.tmpl) file, both for the Web end user and for the PKI administrator. We also describe a simple customization example of the pkiserv.tmpl file.

The z/OS PKI Services user interface display and the certificate-related services that it can provide are the result of the combination of the processes run by the CGI modules and the contents of the template file (pkiserv.tmpl). The CGI modules are not intended to be modified by the user, although modification is possible because these are REXX modules. However, modification requires a thorough understanding of the internals of the modules, which is not expected from most users.

A distinction is made in this chapter between the CGI modules and template sections intended for PKI Services administration and the ones designed to support interaction with the Web end user.

4.1 The template file, CGI, and the Web end user

In this section we look at the structure of the template file and CGI to see how they interact with the Web end user.

4.1.1 The template file sections

The template file is a combination of real HTML tags and specific z/OS PKI tags intended for interpretation by the CGI modules. The HTML statements can contain JavaScript for input verification. The template file has several sections: PROLOG, APPLICATION, TEMPLATE, and INSERTS.

The PROLOG section

This section is intended to contain comments explaining main sections, subsections, named fields, and substitution variables. Example 4-1 shows an excerpt of the PROLOG section.

Example 4-1 Excerpt from the PROLOG section

```
# =====
#
# COMPONENT_NAME: pkiserv.tpl
#
# Licensed Materials - Property of IBM
# 5694-A01
# (C) Copyright IBM Corp. 2001, 2002
# Status = HKY7707
#
# =====
#
# Configuration file for interfacing with R_PKIServ. This file may be
# customized as required by the installation. Any line with an '#' in
# column 1 is considered a comment.
#
# Structure:
#
#   The file contains a mixture of true HTML and HTML like tags. The
#   main tags divide the file into sections, APPLICATION, TEMPLATE,
#   and INSERT, where APPLICATION and TEMPLATE may contain various
#   subsections, named fields, and substitution variables as explained
#   below.
#
#   <APPLICATION NAME=appl-name> ... </APPLICATION>
#
#   This section identifies the applications that will make use of
#   PKI Services for Z/OS. The product ships with one application
```


defined, "PKISERV". This section may contain the following subsections:
#

The APPLICATION section

This section describes the main process streams for the delivered PKI Services. It contains subsections that produce certain Web pages, such as the PKI Services Home page. The <APPLICATION> and </APPLICATION> tags surround one APPLICATION section, as shown in Example 4-2.

Example 4-2 APPLICATION section excerpts

```
<APPLICATION NAME=PKISERV>
<CONTENT>
<HTML><HEAD>
<TITLE> Web Based Certificate Generation Application </TITLE>
%%-copyright%%
</HEAD>
<BODY>
<H1>PKI Services Certificate Generation Application</H1>
<p>
<A HREF="/PKIServ/cacerts/cacert.der">Install
  our CA certificate into your browser </A>
<H2>Choose one of the following:</H2>
<ul>
<li><h3>Request a new certificate using a model</h3>
<FORM name=mainform METHOD=GET ACTION="/PKIServ/ssl-cgi/catmpl.rexx">
<p> Select the certificate template to use as a model
<SELECT NAME="Template">
  %%1 Year PKI SSL Browser Certificate%%
    <OPTION>1 Year PKI SSL Browser Certificate
  %%1 Year PKI S/MIME Browser Certificate%%
    <OPTION>1 Year PKI S/MIME Browser Certificate
  .....
</ADMINHEADER>
<ADMINFOOTER>
<p> %%-pagefooter%%
</BODY>
</HTML>
</ADMINFOOTER>
</APPLICATION>
```

The APPLICATION section contains the following subsections:

► **CONTENT**

This subsection contains the HTML to display the PKI Services Web home page to the end user who is requesting and retrieving certificates. This subsection should contain one or more named fields identifying certificate

templates to use for requesting or managing certificates through this application.

The certificate templates (that is, the information that describes both the interaction with the end user and the internal processes) are described in “The TEMPLATE section” on page 141.

A named field is delineated with %% and is a reference to another part of the template file. Its meaning depends on the section in which it is used:

- ▶ A named field such as %%1 Year PKI SSL Browser Certificate%% is used in the CONTENT subsection to designate a certificate template.
- ▶ A named field such as %%-pagefooter%% is a reference to an INSERT section. An INSERT section is a way to specify common HTML code, such as a common input field or a page header or footer, that must be inserted into a Web page.

See “Named fields” on page 147 for more information about named field.

▶ RECONTENT

The RE in RECONTENT stands for RENEW/REVOKE. This subsection actually is a content subsection that contains the HTML to display information about the client certificate so that the end user can confirm that this is the correct certificate to renew or revoke. This subsection uses the substitution variable [printablecert], which contains the data extracted from the ICL entry.

A substitution variable holds a value that HTML code can reference. At run time, the actual value replaces a substitution variable. A substitution variable is delineated by square brackets. Its value is provided by the CGI code so until it is given a meaningful value by the CGI processes, any reference to it in the template file yields the null string value. “The substitution variables” on page 149 has more about substitution variables.

▶ RESUCCESSCONTENT

This subsection contains the HTML to display a Web page to the end user when the renewal or revocation request is successful. Any named fields in this subsection are interpreted as HTML content inserts (for example, a page footer) that INSERT sections define.

▶ REFAILURECONTENT

This subsection contains the HTML to display a Web page to the end user when a renewal or revocation request is unsuccessful. Any named fields in

this subsection are interpreted as content inserts (for example, a page footer) that INSERT sections define.

► **ADMINHEADER**

This subsection contains the general installation-specific HTML content for the header of all administration Web pages.

► **ADMINFOOTER**

This subsection contains the general installation-specific HTML content for the footer of all administration Web pages.

The INSERT sections

These sections contain HTML to be inserted in certain Web pages (for example, the Request submitted successfully Web page) and certificate field dialogs such as text entry boxes (the common name INSERT produces a text box where the user enters this information) and pull-downs. Example 4-3 shows the INSERT section for the %%CommonName%% named field.

Example 4-3 Sample INSERT section

```
<INSERT NAME=CommonName>
<p> Common Name [optfield] <BR>
<INPUT NAME="CommonName" TYPE="text" SIZE=64 maxlength="64">
</INSERT>
```

The TEMPLATE section

These are the certificate templates (models) that contain the HTML to produce certificate request forms and to retrieve the signed certificates. They also define the fields that are permissible in the certificate. The certificate templates delivered with z/OS PKI Services, as of the writing of this book, are described in “The certificate templates delivered in the z/OS PKI Services” on page 149. Example 4-4 shows some excerpts from a TEMPLATE section.

Example 4-4 Certificate template excerpts: CONTENT subsection

```
<TEMPLATE NAME=1 Year PKI SSL Browser Certificate>
<TEMPLATE NAME=PKI Browser Certificate>
<NICKNAME=1YBSSL>
<CONTENT>
<HTML><HEAD>
<TITLE> Web Based PKIX Certificate Generation Application Pg 2</TITLE>
%%-copyright%%
%%-AdditionalHead[browsertype]%%
<SCRIPT LANGUAGE="JavaScript">
<!--
function ValidateEntry(){
```

```

.....
</SCRIPT>
</HEAD>
<BODY>
<H1>1 Year SSL Browser Certificate</H1>
<p>
<H2>Choose one of the following:</H2>
<p>
<ul>
<h3><li>Request a New Certificate</h3>
<FORM NAME="CertReq" METHOD=POST ACTION=
        "/PKIServ/ssl-cgi-bin/careq.rexx" onSubmit=
        "if(ValidateEntry()) return false; else return true;">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tplname]">
<p> Enter values for the following field(s)
    %%CommonName%%
    %%Email (optional)%%
    %%Requestor (optional)%%
    %%NotifyEmail (optional)%%
    %%PassPhrase%%
    %%PublicKey2[browsertype]%%
<INPUT TYPE="reset" VALUE="Clear">
</FORM>
<p>
<H3><li>Pick Up a Previously Issued Certificate</H3>
<FORM METHOD=GET ACTION="/PKIServ/ssl-cgi/caretrieve.rexx">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tplname]">
<INPUT TYPE="submit" VALUE="Retrieve your certificate">
</FORM>
</ul>
<p>%%-pagefooter%%
</BODY>
</HTML>
</CONTENT>

```

The **TEMPLATE** section can have the following subsections:

- The **CONTENT** subsection

This subsection contains the HTML to display a Web page to the end user who requests a certificate of a specific type. Field names on the certificate request (such as a text box where the user enters a value for Common Name) match the names of **INSERT** sections.

Example 4-4 on page 141 shows an example of a certificate template **CONTENT** subsection. Note that in this example a runtime logic is introduced in the template using JavaScript. Note also the named fields referring to the contents of the certificate fields.

► The CONSTANT subsection

This subsection identifies certificate fields that have a constant (hard-coded) value for everyone. This subsection should contain only named fields, one per line. This is shown in Example 4-5.

Example 4-5 Certificate template excerpts: CONSTANT subsection

```
</CONTENT>
<CONSTANT>
%%NotBefore=0%%
%%NotAfter=365%%
%%KeyUsage=handshake%%
%%OrgUnit=Class 1 Internet Certificate CA%%
%%Org=The Firm%%
%%SignWith=PKI:%%
</CONSTANT>
```

► The ADMINAPPROVE subsection

This optional subsection contains the named fields that the administrator can modify when approving certificate requests. (The named fields refer to INSERT sections.) When an end user requests a certificate, the certificate request may contain fields that the end user cannot see. When approving a request, the administrator can modify:

- Fields that are present and visible to the end user in the certificate request (for example, Common Name).
- Fields that are not visible to the end user but are hardcoded (in the CONSTANT subsection) in the template (such as Organizational unit).
- Fields that are not visible to the end user and that the PKI Services administrator can add, such as HostIdMappings extension or an empty Organizational Unit field. (These are listed in the <ADMINAPPROVE> section, and either the end user did not fill them in or they are not present on the template request form).

The presence of the ADMINAPPROVE subsection (even if empty) indicates that an administrator must approve this request. The absence of this section indicates that this certificate type will be auto-approved.

Example 4-6 shows an ADMINAPPROVE subsection.

Example 4-6 Certificate template excerpts: ADMINAPPROVE subsection

```
</CONSTANT>
<ADMINAPPROVE>
%%CommonName (Optional)%%
%%OrgUnit (Optional)%%
```

```

%%OrgUnit (Optional)%%
%%Org (Optional)%%
%%NotBefore (optional)%%
%%NotAfter (Optional)%%
%%KeyUsage (Optional)%%
%%HostIdMap (Optional)%%
%%HostIdMap (Optional)%%
%%HostIdMap (Optional)%%
%%HostIdMap (Optional)%%
</ADMINAPPROVE>

```

► The SUCCESSCONTENT subsection

This subsection contains the HTML to display to the end user a Web page saying that the certificate request was submitted successfully. Any named fields in this subsection are interpreted as content inserts defined by INSERT sections.

<SUCCESSCONTENT> contains only the named field %%-requestok%%, which contains HTML for the Web page *Request submitted successfully*.

► The FAILURECONTENT subsection

This subsection contains the HTML to display to the end user a Web page stating that the certificate request was not submitted successfully. Any named fields in this subsection are interpreted as content inserts defined by INSERT sections.

<FAILURECONTENT> contains only the named field %%-requestbad%%, which contains HTML for the Web page *Request was not successful*.

Example 4-7 shows an example of the SUCCESSCONTENT and FAILURECONTENT subsections.

Example 4-7 Excerpts: SUCCESSCONTENT and FAILURECONTENT subsections

```

</ADMINAPPROVE>
<SUCCESSCONTENT>
  %%-requestok%%
</SUCCESSCONTENT>
<FAILURECONTENT>
  %%-requestbad%%
</FAILURECONTENT>

```

► The RETRIEVECONTENT subsection

This subsection contains the HTML to display to the end user a Web page to enable certificate retrieval. Any named fields in this subsection are interpreted as content inserts that the INSERT sections define.

<RETRIEVECONTENT> contains:

- The named field %%-copyright%%, which displays any copyright information.
- The title of the Web page. This appears in the banner of your browser.
- A JavaScript script for processing the entry fields on the Web page.
- A heading that says Retrieve Your *(name of certificate)*. This uses the substitution variable [tplname].
- Text: a heading and paragraph about bookmarking this Web page.
- The named field %%TransactionId%% where you enter your transaction ID if it is not already displayed.
- A field where you enter the passphrase you entered on the certificate request form.

Example 4-8 shows content from the RETRIEVECONTENT subsection.

Example 4-8 Examples from the RETRIEVECONTENT subsection

```
<RETRIEVECONTENT>
<HTML><HEAD>
%%-copyright%%
<TITLE> Web Based PKIX Certificate Generation Application Pg 3</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
function MissingTransIdAlert(){
var STRING_MissingTransIdPrompt=
    "Enter the transaction ID assigned to the certificate.";
if(document.retrieveform.TransactionId.value==""){
    alert(STRING_MissingTransIdPrompt);
    document.retrieveform.TransactionId.focus();
    return true;
}
else {
    return false;
}
}
//-->
</SCRIPT>
</HEAD>
.....

<BODY>
<H1> Retrieve Your [tplname]</H1>
<H3>Please bookmark this page</h3>
<p>Since your certificate may not have been issued yet, we recommend
that you create a bookmark to this location so that when you return to
this bookmark, the browser will display your transaction ID.
```

```

This is the easiest way to check your status.
<FORM NAME=retrieveform METHOD=POST ACTION=
    "/PKIServ/ssl-cgi-bin/cagetcert.rexx" onSubmit=
        "if(MissingTransIdAlert()) return false; else return true;">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tplname]">
    %%TransactionId%%
    %%ChallengePassPhrase (optional)%%
<p>
<INPUT TYPE="submit" VALUE="Retrieve and Install Certificate">
</FORM>
<p>
<H2>To check that your certificate installed properly, follow the
procedure below:</h2>
<p><B>Netscape V6</B> - Click Edit->Preferences, then Privacy and Security->
Certificates. Click the Manage Certificates button to start the Certificate
Manager.
Your new certificate should appear in the Your Certificates list.
Select it then click View to see more information.
<p><B>Netscape V4</B> - Click the Security button, then Certificates->
Yours. Your certificate should appear in the list. Select it then
click Verify.
<p><B>Internet Explorer V5</B> - Click Tools->Internet Options, then
Content, Certificates.
Your certificate should appear in the Personal list. Click Advanced to
see additional information.
<p>
<FORM METHOD=GET ACTION="/PKIServ/public-cgi/camain.rexx">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tplname]">
<INPUT TYPE="submit" VALUE="Home page">
</FORM>
<p>%%-pagefooter%%
</BODY>
</HTML>
</RETRIEVECONTENT>

```

► The RETURNCERT subsection

This subsection contains the HTML to display to the end user a Web page upon successful certificate retrieval. For PKISERV, if the certificate being retrieved is a browser certificate, then this section must contain a single line containing a browser-qualified INSERT name. (See Example 4-9.)

Example 4-9 Certificate template excerpts: RETURNCERT subsection

```

</RETRIEVECONTENT>
<RETURNCERT>
%%returnbrowsercert[browsertype]%%
</RETURNCERT>
</TEMPLATE>

```

Additionally, INSERTs for Netscape (returnbrowsercertNS) and Internet Explorer (returnbrowsercertIE) containing browser-specific HTML for returning certificates must be defined elsewhere in the pkiserv.tmpl certificates template file. If the certificate being retrieved is a server certificate, this section should contain the HTML necessary to present the certificate to the user as text.

► The APPL subsection

This subsection identifies certificate fields for which the application itself should provide values. This subsection should contain only named fields, one per line. The only supported named fields allowed in this section are:

- UserId
- HostIdMap

Example 4-10 shows an APPL subsection. As of this writing, the subsection is found only in the certificate templates pertaining to the following certificates:

- SAF Server Certificate 1 Year (Auto-Approved)
- SAF Browser Certificate 1 Year (Auto-Approved)
- 2 Year Browser Certificate For Authenticating to z/OS
- 5 Year PKI Intermediate CA Certificate

Example 4-10 Certificate template excerpts: APPL subsection

```
<APPL>
  %%UserId%%
</APPL>
```

Additional information about the template file contents

Named fields

The following named fields refer to certificate templates (the %% delineator has been removed in this list):

- 1 Year PKI SSL Browser Certificate
- 1Year PKI S/MIME Browser Certificate
- 2 Year PKI Browser Certificate For Authenticating To z/OS
- 5 Year PKI SSL Server Certificate
- 5 Year PKI IPSEC Server (Firewall) Certificate
- 5 Year PKI Intermediate CA Certificate
- 1 Year SAF Browser Certificate
- 1 Year SAF Server Certificate
- PKI Browser Certificate
- PKI Server Certificate
- SAF Browser Certificate
- SAF Server Certificate

The following named fields correspond to INSERT sections in the Template File:

- ▶ -AdditionalHeadIE
- ▶ -requestok
- ▶ -requestbad
- ▶ -renewrevokeok
- ▶ -renewrevokebad
- ▶ -returnpkcs10cert
- ▶ returnbrowsercertNS
- ▶ returnbrowsercertIE
- ▶ KeyUsage
- ▶ NotBefore
- ▶ NotAfter
- ▶ Country
- ▶ Org
- ▶ OrgUnit
- ▶ OrgUnit2
- ▶ Locality
- ▶ StateProv
- ▶ CommonName
- ▶ Title
- ▶ AltIPAddr
- ▶ AltEmail
- ▶ AltURI
- ▶ AltDomain
- ▶ Street
- ▶ PostalCode
- ▶ Email
- ▶ SignWith
- ▶ PublicKey
- ▶ PublicKeyNS
- ▶ PublicKey2NS
- ▶ PublicKeyIE
- ▶ PublicKey2IE
- ▶ UserId
- ▶ Label
- ▶ Requestor
- ▶ PassPhrase
- ▶ ChallengePassPhrase
- ▶ HostIdMap
- ▶ TransactionId
- ▶ NotifyEmail
- ▶ -copyright
- ▶ -pagefooter

The substitution variables

- ▶ **base64cert**
The requested certificate, base64-encoded.
- ▶ **browsertype**
A special substitution variable to qualify named fields only. It enables the different browsers, Netscape and Internet Explorer, to perform browser-specific operations such as generating a public and private key pair. To do this, Netscape uses a KEYGEN HTML tag while Internet Explorer uses ActiveX controls.

For example, suppose you specify %%PublicKey [browsertype]%% in a TEMPLATE CONTENT section. If the user referencing this section uses the Netscape Navigator browser, then INSERT PublicKeyNS is included. If the user's browser is Microsoft® Internet Explorer, INSERT PublicKeyIE is included.
- ▶ **iecert**
The requested certificate in a form that Microsoft Internet Explorer accepts.
- ▶ **optfield**
A special substitution variable that should be placed in any certificate field name INSERT where the end user can supply the value. It makes the input field optional.
- ▶ **printablecert**
This contains the certificate details so that the end user can confirm that the certificate is the correct one to renew or revoke. The displayed data is extracted from the ICL entry.
- ▶ **tplname**
A certificate template name. This is primed from the HTML tag <SELECT NAME="Template"> in the <APPLICATION NAME=PKISERV> section. The end user selects it on the first Web page.
- ▶ **transactionid**
A unique value returned from a certificate request.

The certificate templates delivered in the z/OS PKI Services

- ▶ **1 Year SAF Server Certificate**
The template enables end users to request certificates for servers using native SAF certificate generation facilities (that is, using the RACDCERT GENCERT facility as opposed to the PKI Services GENCERT certificate generation facilities). The certificate is used for handshaking only (for

example, SSL). This certificate is auto-approved, as the requestor is asked for an RACF valid user ID and password.

- ▶ 1 Year SAF Browser Certificate

This template is for requesting a browser certificate. SAF certificate generation facilities (RACDCERT GENCERT) create the certificate. The requestor must input a label because the certificate is stored in an RACF database. This certificate is auto-approved, as the requestor is asked for an RACF valid user ID and password.

- ▶ 1 Year PKI SSL Browser Certificate

A template for requesting a browser certificate that PKI Services generates. The end user enters the common name. This template contains an ADMINAPPROVE section. Therefore, certificates requested using this template require administrator approval before being issued. The user ID and password are not required but the passphrase is required.

- ▶ 1 Year PKI S/MIME Browser Certificate

A template for requesting a browser certificate that PKI Services generates. This is similar to the 1 Year PKI SSL Browser Certificate but contains an AltEmail extension field.

- ▶ 2 Year PKI Browser Certificate For Authenticating To z/OS

A template for requesting a browser certificate that PKI Services generates. This is similar to the one-year PKI SSL browser certificate but includes the %%HostIdMap%% INSERT. This certificate is auto-approved, as the requestor is asked for an RACF valid user ID and password.

- ▶ 5 Year PKI SSL Server Certificate

A template for requesting a server certificate that PKI Services generates. This is similar to the SAF server template except that this template contains an ADMINAPPROVE section. Therefore, certificates requested using this template require administrator approval before being issued. The user ID and password are not required but the passphrase is required.

- ▶ 5 Year PKI IPSEC Server Certificate

A template for requesting a server certificate that PKI Services generates. This is similar to the 5 Year PKI SSL Server Certificate except that keyusages of handshake and dataencrypt are hard-coded. Also, the certificate contains AltEmail, AltIPAddr, AltURI, and AltDomain extension fields.

- ▶ 5 Year PKI Intermediate CA Certificate

A template for requesting a server certificate that PKI Services generates. This is similar to the PKI SSL server template except that KeyUsage is hardcoded as certsign. Also, this certificate is auto-approved because it runs under the user ID of the requestor (that is, the person requesting this must be

highly authorized). The user ID and password are required, and the units of work should run under the client's ID. In other words, the end user must be someone who can do this using RACDCERT alone, that is, must have CONTROL authority to IRR.DIGTCERT.GENCERT, and so forth. Given this requirement, the administrator need not approve this. The PassPhrase is required.

4.1.2 The CGI modules

Here is a description of the CGI modules involved in the interaction with the Web end user. Note that all of these modules are not located in the same directory. Here are their complete paths:

- ▶ /usr/lpp/pkiserv/PKIServ/public-cgi/camain.rexx
- ▶ /usr/lpp/pkiserv/PKIServ/ssl-cgi-bin/catmpl.rexx
- ▶ /usr/lpp/pkiserv/PKIServ/ssl-cgi-bin/auth/careq.rexx
- ▶ /usr/lpp/pkiserv/PKIServ/ssl-cgi-bin/caretrieve.rexx
- ▶ /usr/lpp/pkiserv/PKIServ/ssl-cgi-bin/auth/cagetcert.rexx
- ▶ /usr/lpp/pkiserv/PKIServ/clientauth-cgi-bin/cadisplay.rexx
- ▶ /usr/lpp/pkiserv/PKIServ/clientauth-cgi-bin/camodify.rexx

They perform the following functions:

- ▶ camain.rexx
 - Clicking the Request certificate button calls catmpl.rexx, passing it a parameter identifying the selected template.
 - Clicking the Pick up certificate button goes directly to caretrieve.rexx (if the certificate is already requested).
 - Clicking the Renew or revoke certificate button goes to cadisplay.rexx.
 - An administrator can click the Go to administration page button to go to admmain.rexx.
- ▶ catmpl.rexx
 - Displays Web page coded in the HTML under the CONTENT subsection (of a TEMPLATE section).
 - Clicking the Submit certificate request button passes template and field name parameters to careq.rexx.
 - Clicking the Retrieve your certificate button passes control to caretrieve.rexx.
- ▶ careq.rexx
 - Processes field names under the APPL subsection (of a TEMPLATE section).

Note: Depending on the template, this can be:

- ▶ UserId only
- ▶ UserId and HostIdMap

Note: Processes hard-coded field names under the CONSTANT subsection (of a TEMPLATE section).

- Depending on the results, displays Web page coded in the HTML under the SUCCESSCONTENT or FAILURECONTENT subsection (of a TEMPLATE section):
 - The SUCCESSCONTENT subsection includes a Continue button the user can click to continue to caretrieve.rexx.
- ▶ caretrieve.rexx
 - Displays Web page coded in the HTML under the RETRIEVECONTENT subsection (of a TEMPLATE section). This HTML prompts the user to enter the transaction ID and a password if the user entered one when requesting the certificate.
 - When the user clicks the Retrieve and install certificate button, this passes the transaction ID parameter to cagetcert.rexx.
- ▶ cagetcert.rexx

Displays Web page coded in the HTML under the RETURNCERT subsection (of a TEMPLATE section). This HTML determines which of the following forms to use when returning the certificate:

 - Base64-encoded certificate (for server certificates)
 - ActiveX object (for Microsoft Internet Explorer browser certificates)
 - Application/x-x509-user-certificate MIME type (for Netscape browser certificates)
- ▶ cadisplay.rexx
 - Displays Web page coded in the HTML under the RECONTENT subsection (of the APPLICATION section).
 - For renewing a certificate, the user fills in the passphrase and clicks the Renew button. For revoking a certificate, the user clicks the Revoke button. Both actions call camodify.rexx.
- ▶ camodify.rexx
 - Displays Web page coded in the HTML under the SUCCESSCONTENT subsection (of a TEMPLATE section) for a successful renewal. The

SUCCESSCONTENT subsection includes a Continue button the user can click to call caretrieve.rexx.

- Displays the Web page coded in HTML under the RESUCCESSCONTENT subsection (of the APPLICATION section) for a successful revocation.

4.1.3 Relationship between CGI modules and Web user templates

Figure 4-1 on page 154 and Figure 4-2 on page 154 graphically describe how CGI modules, template file sections, and subsections interact.

- ▶ The main CGI camain.rexx displays the PKI Services Home Page and waits for the end-user inputs.
- ▶ When the end user sends the request for a certificate, catmpl.rexx is given control and displays the page to collect the information required in this certificate template.
- ▶ careq.rexx builds the certificate request, using constants as specified in the CONSTANT subsection of the certificate template. If ADMINAPPROVE exists in the certificate template, then administrator approval is required.

The SUCCESSCONTENT or FAILURECONTENT subsections indicate what to display subsequently to the certificate creation.

- ▶ Request to retrieve the certificate gives control to caretrieve.rexx, which indicates how to proceed with the information in the RETRIEVECONTENT subsection.
- ▶ Finally, the certificate is returned by cagetcert.rexx as indicated in the RETURNCERT subsection.

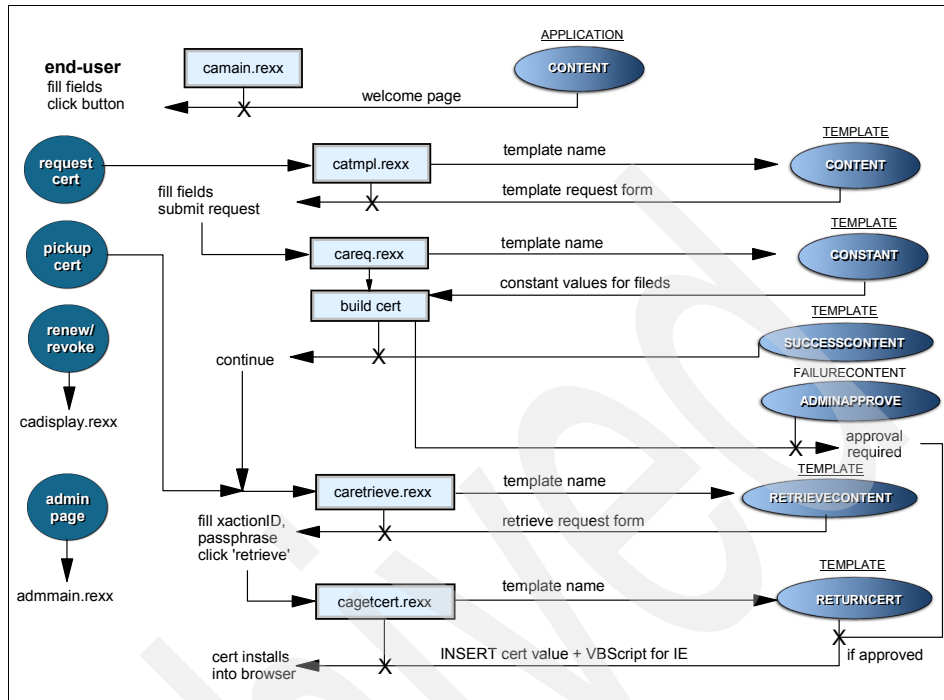


Figure 4-1 Interaction with the Web end user: Part 1 of 2

Figure 4-2 shows more precisely what happens in case of a certificate renewal/revocation request.

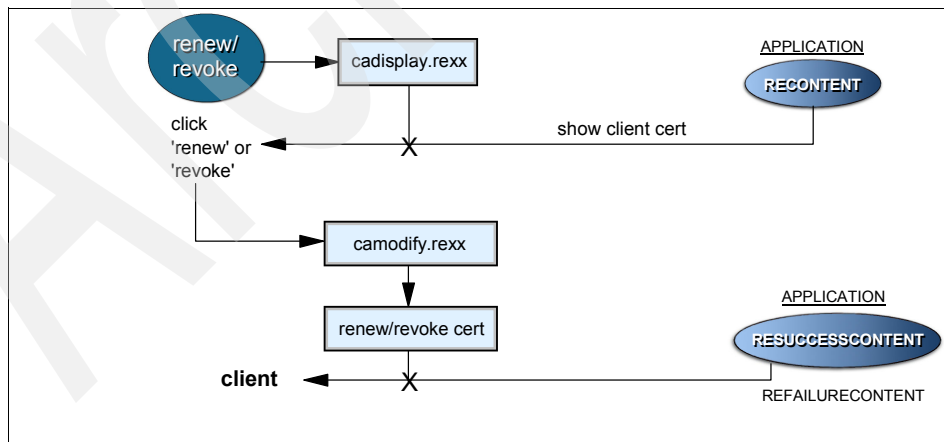


Figure 4-2 Interaction with the Web end user - Part 2 of 2

4.1.4 An example of simple customization of the template file

Additional text in the PKI Services home page

Example 4-11 shows two modifications of the z/OS PKI home page:

- ▶ The page header changes to Web Security Class - z/OS PKI Services Home.
- ▶ The z/OS PKI CA certificate in base 64 encoding is displayed in the end-user browser for a copy-and-paste into another system (as opposed to transparently downloading it into the browser).

Note that this requires additional work in the z/OS PKI Web server. Add to its httpd.conf file the following AddEncoding directive:

```
AddEncoding      .b64      ebcdic
```

Be sure that a copy of the corresponding B64 formatted certificate has been installed in the indicated path.

Example 4-11 Customization of the PKI Services home page

Original:

```
<APPLICATION NAME=PKISERV>
<CONTENT>
<HTML><HEAD>
<TITLE> Web Based Certificate Generation Application </TITLE>
%%-copyright%%
</HEAD>
<BODY>
<H1>PKI Services Certificate Generation Application</H1>
<p>
<A HREF="/PKIServ/cacerts/cacert.der">Install
our CA certificate into your browser </A>
.....
```

Modified:

```
<APPLICATION NAME=PKISERV>
<CONTENT>
<HTML><HEAD>
<TITLE> Web Based Certificate Generation Application </TITLE>
%%-copyright%%
</HEAD>
<BODY>
<H1>Web Security Class - z/OS PKI Services Home</H1>
<p>
<A HREF="/PKIServ/cacerts/cacert.der">Install
our CA certificate into your browser </A>
<p>
```

```
<A HREF="/PKIServ/cacerts/cacert.b64">download  
our CA certificate into your system </A>
```

Inhibiting delivery of some certificates

In our example, we want users to be unable to get some certificate types, but we want to keep them informed that such a facility exists in the z/OS PKI Services. In Example 4-12, we display a short message as a response to a request for a 1 Year SAF Server Certificate.

Note that the HTML Return button delivered in the Template File does not work in that case, as it expects the CGI to have been invoked already before getting a return request. This has been fixed by changing the button to a hyperlink referring to the Home Page:

```
<A href="http://9.100.203.111:8000/demopki/">  
Return to PKI Services Home page</A>
```

Example 4-12 Customization of the PKI Services certificate template

Original:

```
<TEMPLATE NAME=1 Year SAF Server Certificate>  
<TEMPLATE NAME=SAF Server Certificate>  
<CONTENT>  
<HTML><HEAD>  
<TITLE> Web Based SAF Certificate Generation Application Pg 2</TITLE>  
%%-copyright%%  
<SCRIPT LANGUAGE="JavaScript">  
<!--  
function MissingRequiredFAlert(){  
var STRING_MissingRequiredFPrompt=  
    "Enter the field value that's not optional."  
  
if ((document.serverform.Label.value=="")||  
    (document.serverform.OrgUnit.value==""))||  
.....
```

Modified:

```
<TEMPLATE NAME=1 Year SAF Server Certificate>  
<TEMPLATE NAME=SAF Server Certificate>  
<CONTENT>  
<HTML><HEAD>  
<TITLE> Web Based SAF Certificate Generation Application Pg 2</TITLE>  
%%-copyright%%  
</HEAD>  
<BODY>  
<H1> SAF Server Certificate 1 Year (Auto Approved)</H1>  
<p>  
<p>We are sorry - We are not providing this service during
```

this Web Security Class.
 If the service had been available you would have
 been asked for a valid RACF userid and password, and
 your certificate request would have been in turn
 automatically approved.
 <p>Please return to the z/OS PKI Services home page
 <p>

 Return to PKI Services Home page
 <p>

Making the certificates auto-approved

In order to make a certificate type auto-approved, remove the ADMINAPPROVE subsection in the certificate template, as shown in Example 4-13.

Example 4-13 Making a certificate type auto-approved

Original:
 <TEMPLATE NAME=1 Year PKI SSL Browser Certificate>
 <TEMPLATE NAME=PKI Browser Certificate>
 <NICKNAME=1YBSSL>
 <CONTENT>

 <CONSTANT>
 %%NotBefore=0%%
 %%NotAfter=365%%
 %%KeyUsage=handshake%%
 %%OrgUnit=Class 1 Internet Certificate CA%%
 %%Org=The Firm%%
 %%SignWith=PKI:%%
 </CONSTANT>
 <ADMINAPPROVE>
 %%CommonName (Optional)%%
 %%OrgUnit (Optional)%%
 %%OrgUnit (Optional)%%
 %%Org (Optional)%
%
 %%HostIdMap (Optional)%%
 </ADMINAPPROVE>
 <SUCCESSCONTENT>

 Modified:
 <TEMPLATE NAME=1 Year PKI SSL Browser Certificate>
 <TEMPLATE NAME=PKI Browser Certificate>
 <NICKNAME=1YBSSL>
 <CONTENT>


```

</CONSTANT>
#<ADMINAPPROVE>
# %%CommonName (Optional)%%
# %%OrgUnit (Optional)%%
# %%OrgUnit (Optional)%%
# %%Org (Optional)%%
.....
# %%HostIdMap (Optional)%%
#</ADMINAPPROVE>
<SUCCESSCONTENT>

```

4.2 Structure of the template file for interaction with the PKI Administrator

The following section looks at the structure of the template file and how it interacts with the PKI Administrator.

4.2.1 The CGI modules

These modules are all contained in the /usr/lpp/pkiserv/PKIServ/ssl-cgi-bin/auth/ directory:

- ▶ admmain.rexx

This displays the administration home page with the header PKI Services Administration. This Web page enables the administrator to work with a single certificate request or certificate or to search for certificate requests or certificates.

- ▶ admpend.rexx

On the administration home page, the administrator can search for certificate requests. This displays a Web page with one of the following headers:

- Certificate Requests lists certificate requests matching the criteria and enables the administrator to process certificate requests.
- Processing was not successful.

- ▶ admpendtid.rexx

On the administration home page, the administrator can enter a transaction ID to work with a single certificate request. This displays a Web page with one of the following headers:

- Single Request lists the certificate request that matches the transaction ID and enables the administrator to process that certificate request.

- Processing was not successful.
- ▶ `admmodtid.rexx`

This displays the Modify and Approve Request page that appears when the administrator decides to modify a request before approving it (on the Single Request page).
- ▶ `admicl.rexx`

On the administration home page, the administrator can search for certificates. This displays a page with one of the following headers:

 - Issued Certificates lists certificates that match the search criteria and enables the administrator to revoke or delete selected certificates.
 - Processing was not successful.
- ▶ `admiclcert.rexx`

On the administration home page, the administrator can enter a serial number to work with a single certificate. This displays a Web page with one of the following headers:

 - Single Issued Certificate lists the certificate that matches the serial number ID and enables the administrator to revoke or delete that certificate.
 - Processing was not successful.
- ▶ `admacttid.rexx`

Displays a Web page after the administrator processes a single certificate request (approving it with or without modifications, rejecting, or deleting it). This Web page has one of the following headers:

 - Processing successful.
 - Processing was not successful.
- ▶ `admacttid2.rexx`

This displays a Web page after the administrator approves a certificate request with modifications. The Web page has one of the following headers:

 - Processing successful.
 - Processing was not successful.
- ▶ `admpendall.rexx`

After the administrator searches for certificate requests and `admpend.rexx` displays the results, the administrator clicks a button to approve, reject, or delete selected certificate requests. This calls `admpendall.rexx`, whose main header is one of the following:

 - Processing successful if the action was successful.

- Processing was not successful if the action failed (for example, if the administrator tried to delete certificate requests that were already deleted).
 - Processing partially successful if only some of the selected requests are processed successfully.
- **admactcert.rexx**
- After the administrator tries to revoke or delete one or more selected certificates, displays a Web page with one of the following main headings:
- Processing successful.
 - Processing was not successful.
- **admiclall.rexx**
- After the administrator searches for certificates and `admicl.rexx` displays the results, the administrator clicks a button to revoke or delete selected certificates. This calls `admiclall.rexx`, which displays a Web page with one of the following main headings:
- Processing successful if the action was successful.
 - Processing was not successful if the action failed.
 - Processing partially successful if only some of the selected certificates are processed successfully.

4.2.2 Customization of the administration Web pages

The administration Web pages are not as customizable as the end-user Web pages. You can customize page headers, footers, frames, links, colors, and so forth, but you cannot change internal Web page content. Except for identifying the fields that an administrator can change when approving certificate requests (in the `ADMINAPPROVE` subsection of the certificate template), the administration Web page logic is fixed.

However, you can make changes in these two subsections in the `APPLICATION` section of the `pkiserv.tmpl` certificate template file:

- **ADMINHEADER**
Contains the general installation-specific HTML content for the header of all administration Web pages.
- **ADMINFOOTER**
Contains the general installation-specific HTML content for the footer of all administration pages.

Example 4-14 on page 161 shows the specification of `ADMINHEADER` and `ADMINFOOTER` in the template file delivered with the z/OS PKI.

Example 4-14 ADMINHEADER and ADMINFOOTER in the APPLICATION section

```
<ADMINHEADER>
<HTML><HEAD>
<TITLE> Web Based Certificate Generation Administration </TITLE>
%%-copyright%%
</HEAD>
<BODY>
</ADMINHEADER>
<ADMINFOOTER>
<p> %%-pagefooter%%
</BODY>
</HTML>
</ADMINFOOTER>
</APPLICATION>
```

4.2.3 PKI administrator e-mail address

This is an example of customizing the e-mail address of the administrator or the Web site that contains your certification policy.

Change the e-mail address at the end of the /web/server1/pkiserv.tmpl file. Search the mailto parameter:

```
<INSERT NAME=-pagefooter>
<A HREF="mailto:trauner@de.ibm.com">
email: PKI-Administrator</A>
</INSERT>
```

4.2.4 PKI Services certification policy

The pkiserv.conf file contains a section called CertPolicy.

Example 4-15 pkiserv.conf section CertPolicy

```
[CertPolicy]
SigAlg1=sha-1WithRSAEncryption
CreateInterval=3m

# When the warning message should be issued. (i.e. the number of days
# or weeks before the certificate expiration date/time). Defaults to never
ExpireWarningTime=4w

TimeBetweenCRLs=1d
CRLDuration=2d

PolicyRequired=F
PolicyCritical=F
```

```
PolicyName1=MyPolicy
Policy1Org=MyOrganization
Policy1Notice1=3
Policy1Notice2=17
UserNoticeText1=This is some very lawyerly statement for the relying party to
read and make decisions based on.
CPS1=http://www.mycompany.com/cps.html
```

All parameters are described in *z/OS Security Server PKI Services Guide and Reference*, SA22-7693. See the advanced customization chapter.

When you start configuring the PKIserver, do not wait too long for your certificates to be created. While we tested, we set the create interval to 10 seconds:

```
CreateInterval=10s
```

Before going into production, you should adjust the other policy parameters. This should be done according to the IETF rules. Visit the IETF Web site and check RFC2459 at:

<http://www.ietf.org>

4.2.5 Link to PKI Services from your home page

For people to access the PKI Services application, you could set up a link at a home page instead of asking the people to enter the complete URL.

The URL in our setup is `http://pki.itso.ibm.com/PKIServ/public-cgi/camain.rexx`. The entry that you should put in your referral home page (probably `index.html`):

```
<hr>
This is the
<a href="/PKIServ/public-cgi/camain.rexx">PKI Services Appl</a>
<br>
```

To make access easier, you can add an `exec` statement in your first Web server to access the PKI Services application by an easier URL, such as `http://pki.itso.ibm.com/PKIServ`.

The `exec` statement in `httpd.conf` is similar to:

```
Exec      /PKIServ/*      /usr/lpp/pkiserv/PKIServ/public-cgi/camain.rexx
```

It should be placed at the end of your mapping rules for PKI Services.

4.2.6 Certificate authentication for administrators

Another change that you might want to introduce is the authentication of the administrators through an RACF-registered certificate and not through user ID and password.

In our setup you can choose to identify yourself by user ID and password instead of presenting a certificate to avoid complications.

If you have a certificate that is not registered with RACF, then the `%%CERTIF%%` directive will fall back to `%%CLIENT%%`, which asks for user ID and password.

The steps to do this are as follows:

1. Set up a standard PKI server environment as described in Chapter 3, “Easy steps to get PKI up and running” on page 105.
2. Create PKI certificates for your administrators.
3. Register these certificates with RACF on the system your PKI is running. This can be done in two ways:
 - Use RACF commands to add a certificate to RACF. See “Backup to a data set” on page 68.
 - Use the RACF Autoregistration Web Application.
4. Change the setup of the Web servers.

Setting up the first Web server for admin certificate checking

The following sequence shows how to set up the first Web server, which runs on ports 80 and 443 and does no client authentication. We list all of the necessary settings for PKI with the admin certificate authentication.

For the admin certificate authentication we use the default URI of `/PKIServ/`.

For the administration following the old user ID and password authentication, we invented a new URL of `/PKIALT1/`.

In this example, we assume that:

- ▶ The Web server runs on ports 80 and 443.
- ▶ The other Web server runs on port 1443.
- ▶ The domain name for other PKI servers is `pki.itso.ibm.com`.
- ▶ SSL setup is finished.
- ▶ The PKI server user ID is `PKISRV`.
- ▶ Web server config files are in `/web/pki2`.

The httpd.envvars setup is exactly as described in 3.3, “Setting up the Web servers for PKI” on page 107, so there is no need for a change.

For performance reasons, we changed the location of the redirect statements in httpd.conf so that they appear before the protection statements:

1. Search for the following lines in httpd.conf:

```
# ===== #
#
#       User authentication and document protection
#
# ===== #
```

2. Scroll a little farther, then add the Protection statements as illustrated:

```
#       The following rules enable anyone who knows your WEBADM
#       password to use the Web Server remote configuration application.
#
Protection IMW_Admin {
    ServerId      IMWEBSRV_Administration
    AuthType      Basic
    PasswdFile    %%SAF%%
    Mask          WEBADM,webadm
}

Protect /admin-bin/* IMW_Admin WEBADM
Protect /Docs/admin-bin/* IMW_Admin WEBADM
Protect /reports/* IMW_Admin WEBADM
Protect /Usage* IMW_Admin WEBADM
```

3. Add the following statements to your httpd.conf:

```
#-----
# Redirect directives for the PKI server
#-----
```

Note: The widened margin and split lines below are only for appearance here. In the configuration file, this must be coded in one line with one or more blanks between the two split parameters.

```
# this directive redirects the admin requests to the client auth Web server on port 1443
Redirect /PKIServ/ssl-cgi/auth/adm*
    https://pki.itso.ibm.com:1443/PKIServ/ssl-cgi-bin/auth/adm*
#
# this directive redirects all other SSL requests https in case they come in non SSL
Redirect /PKIServ/ssl-cgi/* https://pki.itso.ibm.com:443/PKIServ/ssl-cgi-bin/auth/*
#
# this directive redirects all PKIALT1 SSL requests https in case they come in non SSL
Redirect /PKIALT1/ssl-cgi/* https://pki.itso.ibm.com:443/PKIServ/ssl-cgi-bin/auth/*
```

```

#
# this directive redirects the clientauth requests to the client auth Web server on port 1443
Redirect /PKIServ/clientauth-cgi/* https://pki.itso.ibm.com:1443/PKIServ/clientauth-cgi/*
#
# this directive redirects the PKIALT1 clientauth requests to the client auth Web server on
port 1443 in case somebody follows that pattern.
Redirect /PKIALT1/clientauth-cgi/* https://pki.itso.ibm.com:1443/PKIServ/clientauth-cgi/*
#

#-----
# End of Redirect directives for the PKI server
#-----
#-----
# Protection directives for the PKI server
#-----

Protection PublicUser {
    ServerId      PublicUser
    UserID        PKISRV
    Mask          Anyone
}
Protect /PKIServ/public-cgi/*      PublicUser
Protect /PKIServ/ssl-cgi-bin/*     PublicUser
Protect /PKIServ/*                 PublicUser

Protect /PKIALT1/public-cgi/*      PublicUser
Protect /PKIALT1/ssl-cgi-bin/*     PublicUser
Protect /PKIALT1/*                 PublicUser

Protection AuthenticatedUser {
    ServerId      AuthenticatedUser
    AuthType      Basic
    PasswdFile    %%SAF%%
    UserID        %%CLIENT%%
    Mask          All
}
Protect /PKIServ/ssl-cgi-bin/auth/* AuthenticatedUser
Protect /PKIALT1/ssl-cgi-bin/auth/* AuthenticatedUser

Protection SurrogateUser {
    ServerId      SurrogateUser
    AuthType      Basic
    PasswdFile    %%SAF%%
    UserID        PKISRV
    Mask          All
}

Protect /PKIServ/ssl-cgi-bin/surrogateauth/* SurrogateUser
Protect /PKIALT1/ssl-cgi-bin/surrogateauth/* SurrogateUser
#-----
# End of protection directives for the PKI server

```

```
#-----
```

4. Now search for the following lines in httpd.conf:

```
# ===== #  
#  
#      Mapping rules  
#  
# ===== #
```

5. Scroll a little farther, then add the mapping rules as illustrated:

```
# *** ADD NEW PASS RULES HERE ***  
  
#-----  
# Mapping rules for the PKI server  
#-----  
#  
Exec      /PKIServ/public-cgi/*      /usr/lpp/pkiserv/PKIServ/public-cgi/*  
Exec      /PKIServ/ssl-cgi-bin/*      /usr/lpp/pkiserv/PKIServ/ssl-cgi-bin/*  
Exec      /PKIALT1/public-cgi/*      /usr/lpp/pkiserv/PKIServ/public-cgi/*  
Exec      /PKIALT1/ssl-cgi-bin/*      /usr/lpp/pkiserv/PKIServ/ssl-cgi-bin/*  
#  
Pass      /PKIServ/cacerts/* /var/pkiserv/* #<- check your certificate dir  
#  
Exec      /PKIServ/*      /usr/lpp/pkiserv/PKIServ/public-cgi/camain.rexx  
Exec      /PKIALT1/*      /usr/lpp/pkiserv/PKIServ/public-cgi/camain.rexx  
#  
#-----  
# End of mapping rules for the PKI server  
#-----  
# Note that we set up the public Web pages to reside in /web/server1/pub  
Pass      /*      /web/server1/pub/*
```

Setting up the second Web server for admin certificate checking

This is how we set up the second Web server, which runs on port 1443 and does client authentication.

In the following text, we list all necessary settings for PKI with admin certificate authentication.

For admin certificate authentication, we use the default URL of /PKIServ/.

For administration following the old user ID and password authentication, we invented a new URI of /PKIALT1/.

In this example, we assume that:

- ▶ The Web server runs on port 1443.
- ▶ The other Web server runs on ports 80 and 443.
- ▶ The domain name for other PKI servers is pki.itso.ibm.com.
- ▶ SSL setup is finished.
- ▶ The PKI server user ID is PKISRV.
- ▶ Web server config files are in /web/pki2a.

The httpd.envvars setup is exactly as described in 3.3, “Setting up the Web servers for PKI” on page 107, so there is no need for a change.

In this setup we did not change the location of the redirect statements in httpd.conf because one EXEC statement must be processed first.

1. Search for the following lines in httpd.conf:

```
# ===== #
#
#       User authentication and document protection
#
# ===== #
```

2. Scroll a little farther and add the Protection statements as illustrated here:

```
#       The following rules enable anyone who knows your WEBADM
#       password to use the Web Server remote configuration application.
#
Protection IMW_Admin {
    ServerId      IMWEBSRV_Administration
    AuthType      Basic
    PasswdFile    %%SAF%%
    Mask          WEBADM,webadm
}

Protect /admin-bin/* IMW_Admin WEBADM
Protect /Docs/admin-bin/* IMW_Admin WEBADM
Protect /reports/* IMW_Admin WEBADM
Protect /Usage* IMW_Admin WEBADM
```

3. Add the following statements to your httpd.conf:

```
#-----
# Protection directives for the PKI server
#-----
Protection PublicUser {
    ServerId      PublicUser
    UserID        PKISRV
    Mask          Anyone
}
Protect /PKIServ/public-cgi/* PublicUser
Protect /PKIServ/ssl-cgi-bin/* PublicUser
```

```

Protect /PKIServ/*          PublicUser

Protect /PKIALT1/public-cgi/* PublicUser
Protect /PKIALT1/ssl-cgi-bin/* PublicUser
Protect /PKIALT1/*          PublicUser

Protection AuthenticatedUser {
    ServerId      AuthenticatedUser
    AuthType      Basic
    PasswdFile    %%SAF%%
    UserID        %%CLIENT%%
    Mask          All
}
Protect /PKIServ/ssl-cgi-bin/auth/* AuthenticatedUser
Protect /PKIALT1/ssl-cgi-bin/auth/* AuthenticatedUser

Protection SurrogateUser {
    ServerId      SurrogateUser
    AuthType      Basic
    PasswdFile    %%SAF%%
    UserID        PKISERV
    Mask          All
}

Protect /PKIServ/ssl-cgi-bin/surrogateauth/* SurrogateUser
Protect /PKIALT1/ssl-cgi-bin/surrogateauth/* SurrogateUser
#-----
# End of protection directives for the PKI server
#-----

```

4. Now search for the following lines in httpd.conf:

```

# ===== #
#
# Mapping rules
#
# ===== #

```

5. Scroll a little farther and add the mapping rules as illustrated:

```

# *** ADD NEW PASS RULES HERE ***

#-----
# Mapping rules for the PKI server
#-----
#

```

Note: The margins are widened below to prevent splitting lines that should be entered on a single line.

```

Exec      /PKIALT1/clientauth-cgi/*  /usr/lpp/pkiserv/PKIServ/clientauth-cgi-bin/*
Exec      /PKIServ/clientauth-cgi/*  /usr/lpp/pkiserv/PKIServ/clientauth-cgi-bin/*
Exec      /PKIServ/ssl-cgi-bin/auth/*  /usr/lpp/pkiserv/PKIServ/ssl-cgi-bin/auth/*
#-----
# Redirect directives for the PKI server
#-----
# redirect the public requests to the non SSL server
Redirect  /PKIServ/public-cgi/* http://pki.itso.ibm.com:80/PKIServ/public-cgi/*
# redirect the non client auth requests to the other Web server on port 443
Redirect  /PKIServ/ssl-cgi/* https://pki.itso.ibm.com:443/PKIServ/ssl-cgi-bin/*
# redirect the public requests to the non SSL server for the alternate schema
Redirect  /PKIALT1/public-cgi/* http://pki.itso.ibm.com:80/PKIALT1/public-cgi/*
# redirect the non client auth requests to the other Web server on port 443
Redirect  /PKIALT1/ssl-cgi/* https://pki.itso.ibm.com:443/PKIALT1/ssl-cgi-bin/*

#-----
# End of Redirect directives for the PKI server
# End of mapping rules for the PKI server
#-----
# Note that we have set up the public Web pages to reside in
/web/server2/pub
Pass      /*                      /web/server2/pub/*

```

To manage the access to the admin application, we set up some HTML links in our home page. It should look similar to Example 4-16.

Example 4-16 html to link to PKI admin function

```

<html><head>
<title>HTTP Server 5.3 powered by z/OS </title>
</head><body bgcolor="#FFFFFF">
<H1>Welcome to the PKI2 WebServer running on z/OS </H1>
<hr>
This is the
<a href="/PKIServ/public-cgi/camain.rexx"> PKI Services Appl</a>
<br>
Directly access the
<a href="/PKIServ/ssl-cgi/auth/admmain.rexx"> PKI administrator</a>
functions using a SSL client certificate.
<br>
Directly access the
<a href="/PKIALT1/ssl-cgi/auth/admmain.rexx"> PKI administrator</a>
functions using RACF user ID and password.
<br>
This is the
<a href="/rar/selfreg.html"> RACF Selfreg Appl</a>
Follow this link to access the
<br>

```

```
<a href="/admin-bin/webexec/cfgstart.html"> Server Administration</a>
</html>
```

Accessing that home page produces a window similar to Figure 4-3.

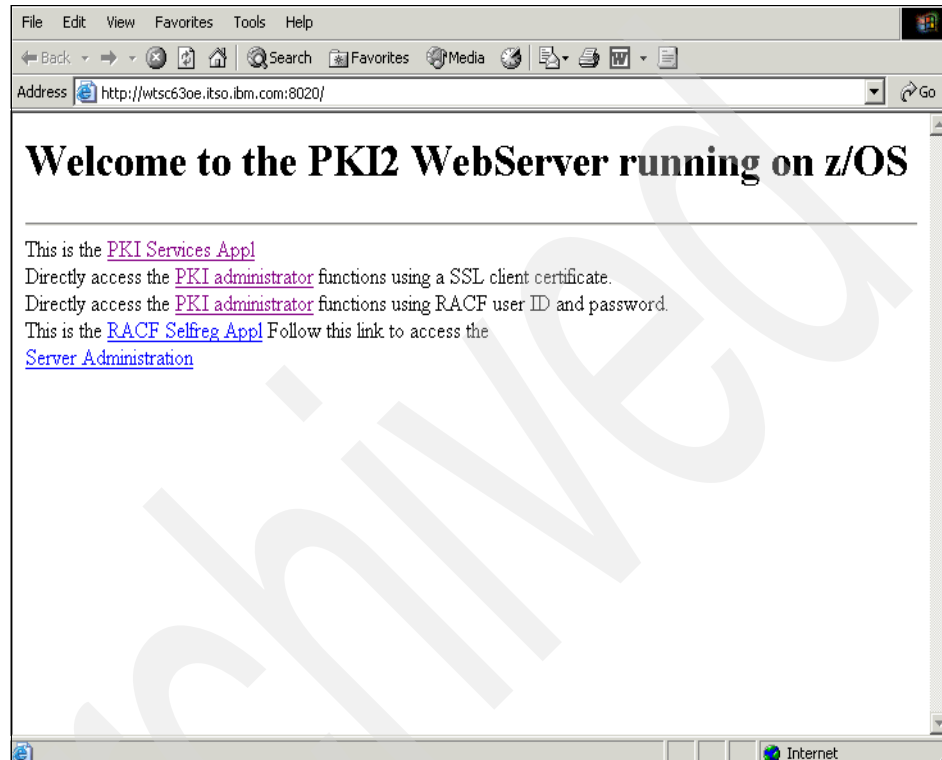


Figure 4-3 PKI Web server home page

Assuming that you have a client certificate ready in your browser, you can use the access for SSL client certificates.

If the certificate is registered in RACF and this user ID is authorized for PKI administration, then you will be able to access the PKI admin function without providing an RACF user ID and password.

Click on the link, and the browser sends the certificate to the Web server. If you have more than one certificate, the browser prompts for the certificate to be used, as shown in Figure 4-4.

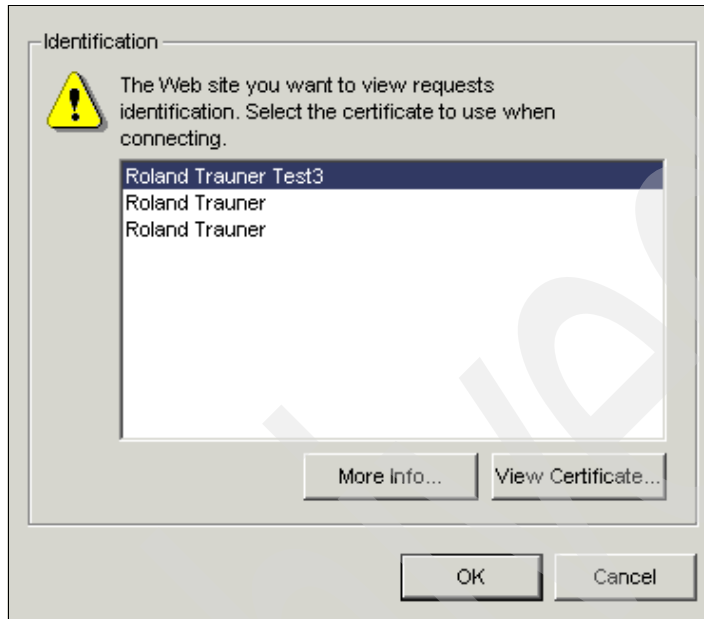


Figure 4-4 Internet Explorer prompt for certificates

If the certificate and the RACF setup are valid, this opens the admin function page shown in Figure 4-5 on page 172.

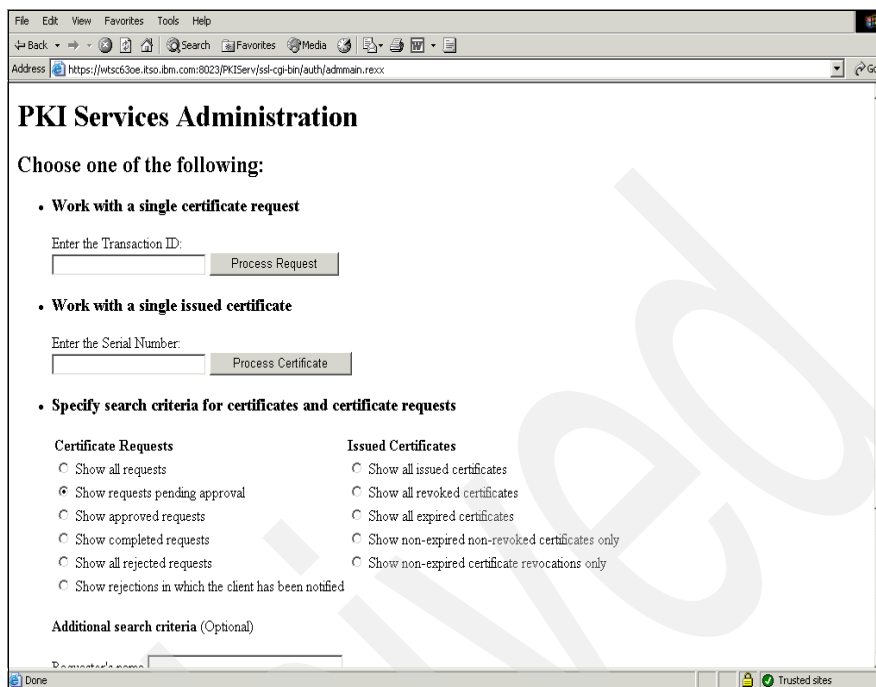


Figure 4-5 PKI Services admin function page

Ensure that the certificate is not revoked (Figure 4-6).

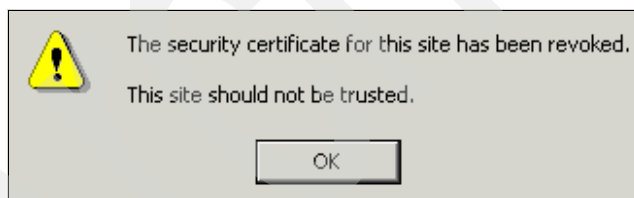
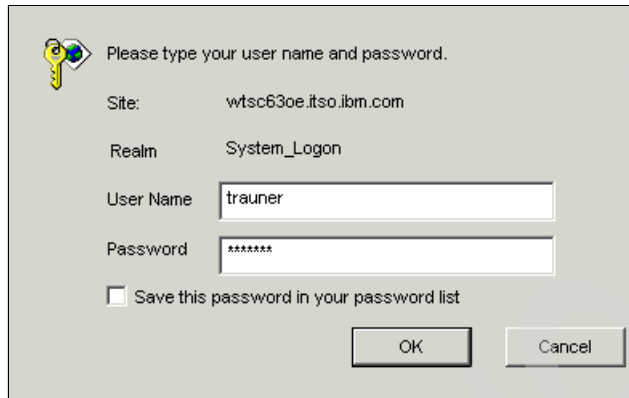


Figure 4-6 Revoked certificate

If the certificate is not registered with RACF, you will be prompted to provide user ID and password as shown in Figure 4-7. User ID and password are always required if you use the access to administer using user ID and password.



Please type your user name and password.

Site: wtsc63oe.itso.ibm.com

Realm: System_Logon

User Name: trauner

Password: *****

☐ Save this password in your password list

OK Cancel

Figure 4-7 Prompt for user ID and password

PKI Installation using the IKYSETUP REXX exec

In this chapter we show how to install PKI Services on z/OS using the IKYSETUP REXX exec, which is provided in SYS1.SAMPLIB(IKYSETUP) and in Chapter 26 of *z/OS Security Server PKI Services Guide and Reference*, SA22-7693. Chapter 5 of this book discusses the modifications required in the original source for the IKYSETUP.

5.1 IKYSETUP overview

IKYSETUP is a REXX exec that performs RACF administration tasks for setting up PKI Services. You would expect that this would be updated and run by the RACF administrator at your site. The exec issues RACF commands to perform the following tasks:

- ▶ Add groups and user IDs.
- ▶ Create access control to protect both the end-user and administrative functions of PKI Services.
- ▶ Create both CA and SSL certificates.
- ▶ Set up the z/OS HTTP server for surrogate operation.

The IKYSETUP exec contains a number of parts, including:

- ▶ The Configurable section, in which values are assigned to variables.
- ▶ The section that issues the RACF commands.
- ▶ A logging section.

Note: The IKYSETUP default is to create a log. Recording information to the log can be disabled by changing `value(log_dsn)`.

The configurable section contains three types:

1. Compulsory changes: Values you must change, such as Company name. See Table 5-1 on page 177 for details.
2. Probable changes (more likely to change): These could be site-specific changes, such as if your site uses ICSF, this would be included in the IKYSETUP exec.
3. Optional changes (unlikely to change): These defaults are acceptable without change.

Important: IKYSETUP should only be updated and run if you have not already done so for an earlier release.

5.2 IKYSETUP variables

This section looks at the IKYSETUP variables. We first look at the compulsory changes, then review the probable changes you will make, and finally consider the optional changes.

5.2.1 Compulsory changes to IKYSETUP

Table 5-1 shows details of the compulsory changes required when updating the IKYSETUP exec. The table includes the modifications we made (in bold under the column Default value, values we specified for our PKI).

Table 5-1 IKYSETUP variables that are compulsory to change

Variable name	Description	Referenced elsewhere	Default value, values we specified for our PKI
ca_dn	The CA's distinguished name. If you already have your certificate and private key set up in RACF, set ca_dn = "", set ca_label to the value of your CA's label, and update ca_expires to reflect the expiration date of your CA certificate. If you do not already have your CA certificate and private key set up in RACF, cross out the default and record your company specific information.	The suffix of the PKI Services CAs must match the LDAP suffix.	OU('Human Resources Certificate Authority') O('Your Company') C('Your Country 2 Letter Abbreviation') ca_dn=, "OU('ITSO PSIE CA')", "O('IBM')", "C('US')"
ca_label	The CA certificate label.	No	Local PKI CA ca_label = "IBM ITSO PSIE PKI1" /* Label for CA certificate */
daemon_uid	The z/OS UNIX user identifier (UID) associated with the PKI Services daemon user ID.	No	554 uid="autoid" /* uid for PKI daemon
pki_gid	The z/OS UNIX group identifier (GID) for the PKI Services administration group.	No	655 gid="autogid" /* PKI Services Admin group id

Variable name	Description	Referenced elsewhere	Default value, values we specified for our PKI
pkigroup_mem.	<p>Members of the PKI Services administration group are responsible for administering PKI Services functions.</p> <p>Recommendation: Restrict PKI Services administration tasks to those with the RACF SPECIAL attribute. pkigroup_mem. is a list in which pkigroup_mem.0 is the number of members in the list, and the rest of the entries are their user IDs. You must change the pkigroup_mem.0 to at least 1 and change pkigroup_mem.1 through pkigroup_mem.n to the member user IDs.</p>	No	<p>0</p> <p>pkigroup_mem.0=6 /* Number of pkigroup members to connect */ pkigroup_mem.1="antoff" pkigroup_mem.2="tramer" pkigroup_mem.3="ranieri" pkigroup_mem.4="chrisr" pkigroup_mem.5="jones" pkigroup_mem.6="kappele"</p>
surrog_uid	The UID associated with the surrogate user ID.	No	<p>555</p> <p>uid="autoid" /* uid for the surrogate id</p>

Variable name	Description	Referenced elsewhere	Default value, values we specified for our PKI
web_dn	Your Web server's distinguished name.	The value of the Web server's common name (CN), which is your server's symbol IP address	CN('www.YourCompany.com') O('Your Company') L('Your City') SP('Your Full State or Province Name') C('Your Country 2 Letter Abbreviation') web_dn=, "CN('wtsc64oe.itso.ibm.com')", "O('IBM')", "L('Poughkeepsie')", "SP('NewYork')", "C('US')"
web_ring	The name of the Web server's SAF key ring. If your Web server is configured for SSL and you are using an RACF key ring, set web_ring to the value of the RACF key ring. If your Web server is configured for SSL and you are using gskkyman, set web_ring="".	httpd*.conf - KeyFile directive	SSLring web_ring = "SSLRING" /* SAF key ring for web serve

5.2.2 Probable changes to IKYSETUP

Table 5-2 on page 180 shows probable changes. These could be site-specific changes, such as if your site uses ICSF this would be included in the IKYSETUP exec. Again, we have included our modifications (when there were any) in bold under the column Default value, value we specified for our PKI.

Table 5-2 Possible IKYSETUP changes

Variable name	Description	Referenced elsewhere	Default value, value we specified for our PKI
bpx_userid	A list of user IDs with daemon and server authority. The bpx_userid.0 is the number of items in the list, and the rest of the entries are the UNIX user IDs. (This is not applicable if unix_sec<>2.)	No	1 (default number of items) OMVSKERN bpx_userid.0=2 /* Number of additional bpxserveridsbelow*/ bpx_userid.1="OMVSKERN" bpx_userid.2="pkistu"
ca_keysize	This is the size in bits of the certificate authority's private key. Valid values are between 1025 and 2048, or use_icsf=2. This indicates that PCICC will be used.	No	1024
csfkeys_profile	This is a profile that protects the PKI Services key in ICSF. (Not applicable if use_icsf is set to 0.)	No	IRR.DIGTCERT.CERTIFAUTH /*csfkeys_profile="IRR.DIGTCERT.**
csfserv_profile	A profile to protect ICSF services. (Not applicable if use_icsf set to 0)	No	CSF*
csfusers_grp	A group of authorized ICSF service users. (Not applicable if use_icsf set to 0.)	No	

Variable name	Description	Referenced elsewhere	Default value, value we specified for our PKI
key_backup	Specifies whether the PKI Services CA certificate and private key should be backed up to an encrypted data set. Valid Values are: 1 (yes - the default) 2 (no) This value is ignored when use_icsf=2 is specified.	No	1(yes)

Variable name	Description	Referenced elsewhere	Default value, value we specified for our PKI
pgmcntl_dsn.	A list in which pgmcntl_dsn.0 is the number of items in the list and the rest of the entries are a list of load libraries to be program-controlled.	No	9(default for number of items) 'CEE.SCEERUN' 'CBC.SCLBDLL' 'GLD.SGLDLNK' 'GSK.SGSKLOAD' 'SYS1.CSSLIB' 'TCPIP.SEZALOAD' 'SYS1.LINKLIB' 'CSF.SCSFMOD0' 'CSF.SCSFMOD1' pgmcntl_dsn.0=9 /* Number of program controlled data sets below */ pgmcntl_dsn.1=""CEE .SCEERUN"" pgmcntl_dsn.2=""CBC .SCLBDLL"" pgmcntl_dsn.3=""GLD .SGLDLNK"" pgmcntl_dsn.4=""GSK .SGSKLOAD"" pgmcntl_dsn.5=""SYS 1.CSSLIB"" pgmcntl_dsn.6=""TCP IP.SEZALINK"" pgmcntl_dsn.7=""SYS 1.LINKLIB"" pgmcntl_dsn.8=""CSF .SCSFMOD0"" pgmcntl_dsn.9=""CSF .SCSFMOD1""

Variable name	Description	Referenced elsewhere	Default value, value we specified for our PKI
restrict_surrog	This specifies whether the surrogate user ID should be restricted. Valid values are: 0(no default) 1(yes) Its is recommended that the default be left the first time IKYSETUP is run but changed before going into a production environment.	No	0(no)
unix_sec	Specifies whether to set up z/OS UNIX level security. Valid values are: 0 (do not set up - default) 1 (is already set up) 2 (add this level of security) For unix_sec equal to 1 or 2, a review of the bpx_userid. and pgmcntl_dsn. rows is required.	For unix_sec=2, the names of the load libraries must change.	0(no) unix_sec=2
use_icsf	Specifies whether PKI Services should use ICSF and PCICC for private key operations. Valid values are: 0(no default) 1 (use ICSF but not PCICC) 2 (use ICSF and PCICC)	If choosing option 1 or 2, ICSF must be configured for RSA(PKA) operations and running.	0(no)

5.2.3 Optional changes to IKYSETUP

Table 5-3 lists the optional updates.

Table 5-3 Optional changes to IKYSETUP

Variable name	Description	Referenced elsewhere	Default value, values we set for our PKI
backup_dsn	This is the data set that contains a backup copy of the PKI Services certificate and private key.	No	'daemon.PRIVATE.KEY.BACKUP.P12BIN' backup_dsn = "PKI.CAPKI1.BACKUP.P12BIN"
ca_expires	The date the PKI Services CA certificate expires.	No	2020/01/01 ca_expires = "2010/01/01"
ca_ring	The name of the PKI Services SAF key ring.	pkiserv.conf - (SAF) KeyRing value	CARing ca_ring="CARING"
daemon	PKI Services daemon user ID.	pkiserv.conf - (SAF) keyRing value	PKISRVD daemon="PKISTU"
export_dsn	The data set that contains the PKI Services certificate for copying to HFS.	No	'daemon.PRIVATE>CA.CERT.DERBIN' export_dsn = "PKI.CAPKI1.DERBIN"
log_dsn	Log data set name.	No	'your-id.PRIVATE.IKYSETUP.LOG' log_dsn="ANTOFF.IKYSETUP.LOG"
pkigroup	PKI Services administration group. This is an RACF group that contains a list of user IDs that are authorized to use PKI Services administration functions.	No	PKIGRP pkigroup="PKIADM" /

Variable name	Description	Referenced elsewhere	Default value, values we set for our PKI
surrog	Surrogate user ID for PKI Services. This cannot be an existing user ID. (IKYSETUP creates the user ID with the NOPASSWORD attribute.)	httpd*.conf - Surrogate user ID	PKISRV surrog="PKISRV"
vsamhlq	HLQ for the VSAM data sets for PKI Services	pkiserv.conf -(ObjectStore) *DSN values IKYCVSAM - Dataset names	Same as daemon variable vsamhlq="pki"
web_expires	Date the Web server certificate expires	No	2020/01/01
web_label	Label for the Web servers certificate	No	SSL cert web_label = "SSL CERT"
webserver	Web servers daemon user ID	See Web server documentation.	WEBSRV webserver="WEBSTU"

In Example 5-1, we highlighted changes to the exec that might be popular with the majority of readers. Because the exec does not provide any OWNER's values (these values are site-specific), the OWNER of the newly created RACF profiles is the user ID running the exec. We suggest that these profiles be changed later to have an OWNER according to local policies.

Example 5-1 IKYSETUP REXX

```

***** Top of Data *****
/* REXX */
/*****/
/*
/* DESCRIPTIVE NAME:  PKI Services RACF setup CLIST
/*
/* Licensed Materials - Property of IBM
/* 5694-A01
/* (C) Copyright IBM Corp. 2001
/* Status = HKY7706
/*
/*01* EXTERNAL CLASSIFICATION: OTHER
/*01* END OF EXTERNAL CLASSIFICATION:

```

```

/*                                                                    */
/* FUNCTION:                                                            */
/*                                                                    */
/* This REXX will issue the RACF TSO commands necessary to set up*/
/* security for PKI Services. It must be run from TSO by a user ID*/
/* that is RACF SPECIAL.                                              */
/*                                                                    */
/* USAGE:                                                                */
/*                                                                    */
/* 1) Read accompanying PKI Services post installation                */
/* instructions.                                                       */
/* 2) Perform necessary prerequisite product installation for         */
/* the webserver (websphere), LDAP, etc.                             */
/* 3) Make note of any predetermined values such as the LDAP         */
/* suffix, webserver fully qualified domain name, and the            */
/* settings contained in the pkiserv.conf file.                      */
/* 4) Copy the REXX to a data set where you can edit it.             */
/* 5) Examine the entire REXX, in particular, the configurable       */
/* section.                                                            */
/* 6) Modify the values in the configurable section as needed for    */
/* your installation.                                                  */
/* 7) Run the REXX. Syntax:                                           */
/*                                                                    */
/* EX 'data-set-name(IKYSETUP)' 'RUN(YES | NO | PROMPT)'              */
/*                                                                    */
/* where: YES - indicates to run REXX as is                           */
/*         NO - indicates to display the commands only               */
/*         PROMPT - indicates to prompt the user prior               */
/*         to invoking each command                                  */
/*                                                                    */
/* DISCLAIMER:                                                          */
/*                                                                    */
/* This REXX is not intended to cover every possible customer        */
/* scenario. Modification of the actual commands to be issued        */
/* may be required                                                     */
/*                                                                    */
/*****/
trace value('0')
/*-----*/
/* configurable section                                              */
/*-----*/
/*-----*/
/* Part 1 - Things you must change */
/*-----*/
/*****/
/* This exec will create the certificate, private key, and          */
/* keyring needed for your certificate authority.                    */
/*                                                                    */
/* You must update the distinguished name of your certificate      */

```



```

/* authority defined below. The suffix of this DN must match */
/* the suffix set up for your LDAP directory (suffix value from */
/* your slapd.conf file). */
/* */
/* Typically, Certificate Authorities have distinguished names */
/* in the following form: */
/* */
/* OU=<your-CA's-friendly-name>,O=<your-organization>, */
/* C=<your-2-letter-country-abbreviation> */
/* */
/* e.g., OU=Human Resources Certificate Authority.O=IBM,C=US */
/* */
/* If you already have your CA certificate and private key set */
/* up in RACF, set ca_dn="" and update the ca_label variable to */
/* equal your CA certificate's label. Note, it must reside */
/* under CERTAUTH */
/* */
/*****/
ca_dn=,
    "OU('ITSO PSIE CA')",
    "O('IBM')",
    "C('US')"
ca_label = "IBM ITSO PSIE PKI1" /* Label for CA certificate */

/* Change 1 */
/*****/
/* This exec will create the certificate, private key, and */
/* keyring needed for your webserver. (Required for SSL.) */
/* */
/* You must update the distinguished name of your */
/* webserver. The Common Name (CN) must match your webserver's */
/* fully qualified domain name. */
/* */
/* e.g., CN=www.ibm.com,O=IBM,C=US */
/* */
/* If you already have your webserver configured for SSL, set */
/* web_dn="" */
/*****/
web_dn=,
    "CN('wtsc64oe.itso.ibm.com')",
    "O('IBM')",
    "L('Poughkeepsie')",
    "SP('New York')",
    "C('US')"

/* Change 2*/
/*****/
/* The sample web server protection directives supplied by PKI */
/* use SSLring for the web server's SAF key ring. If you change */
/* the value below, you will need to modify the "KeyFile" */

```

```

/* directive in the samples/httpd.conf and samples/httpd2.conf */
/* files when configuring the web server. */
/* */
/* If you already have your webserver configured for SSL and */
/* are using a SAF key ring (vs a gskkyman keyfile), then set */
/* web_ring equal to your webserver's SAF key ring name. If you */
/* are using a gskkyman keyfile, then set web_ring="". Note, */
/* you will have to add the CA's certificate to the webserver's */
/* keyfile manually */
/* */
/*****
    web_ring = "SSLRING"                /* SAF keyring for web server */

/* Change 3 */
/*****
/* You must provide UID and GID values for the user IDs and */
/* groups being created below */
/*****
    daemon="PKISTU"                    /* user ID for PKI daemon */
    uid="autoid"                      /* uid for PKI daemon */
    surrog="PKISRV" /* user ID for the surrogate */
    uid="autoid"                      /* uid for the surrogate id */

/* change 4 */
/*****
/* pkigroup members are authorized to administer PKI Services */
/* certificates and certificate requests. If you know the user */
/* IDs that should be connected to this group, update the */
/* pkigroup_mem stem variable. If not, you can always connect */
/* users later. */
/* */
/* If you do not wish to have this exec create this group, */
/* set the group name to "" */
/* */
/*****
    pkigroup="PKIADM" /* PKI Services Admin group name */

/* Change 5 */
    gid="autogid" /* PKI Services Admin group id */

/* Change 6 */
    pkigroup_mem.0=6 /* Number of pkigroup members to connect */
    pkigroup_mem.1="antoff"
    pkigroup_mem.2="trauner"
    pkigroup_mem.3="ranieri"
    pkigroup_mem.4="chrisr"
    pkigroup_mem.5="jjones"
    pkigroup_mem.6="kappele"
/*change 7 */
/*-----*/

```

```

/* Part 2 - Questions you must answer */
/*-----*/
/*****
/* Question 1 - Restrict the surrogate user ID? */
/*
/* The surrogate user ID is the identity assigned to client */
/* processes when requesting certificate services. The */
/* RESTRICTED attribute can be assigned to this ID to limit the */
/* resources available to this user should the user ID be */
/* hijacked by an unfriendly client (hacker). We recommend */
/* that you run the surrogate this way. However, this probably */
/* will cause additional setup work. If you want the RESTRICTED */
/* attribute assigned now, set restrict_surrog=1. Note, you */
/* can always do this at some later time. */
/*****
restricted_surrog=0

/*****
/* Question 2 - Use ICSF? */
/*
/* if ICSF key protection is desired for your CA's private key, */
/* set use_icsf=1. ICSF must be configured for PKA support and */
/* running for this to be successful. Note, you can defer this */
/* until later if you wish. Read the next paragraph before */
/* making this decision */
/*****
use_icsf=0

/*****
/* If you set use_icsf=1 above, you will need to restrict access*/
/* to the CA's private key. Unless you indicate otherwise, this */
/* exec will activate the CSFKEYS class, create a profile in the*/
/* CSFKEYS class to protect the CA's private key, and permit */
/* the PKI Services daemon to use it. */
/*
/* If you are already using ICSF, then you may have profiles in */
/* the CSFSERV class protecting ICSF services. The PKI Services */
/* daemon would need access to the profile that covers the */
/* CSFDSV and CSFDSG services. Also, the PKI Services surrogate */
/* ID would need access to the profile that covers the */
/* CSFENC and CSFDEC services. You may also have a RACF group */
/* for authorized ICSF users. Both of these user IDs */
/* would need to be added to this group. */
/*
/* Set the following variables as needed: */
/*
/* csfkeys_profile - Profile to be created in the CSFKEYS class */
/* Set the value to '' if you don't want the profile */
/* csfserv_profile - Profile to be created in the CSFSERV class */

```

```

/*      e.g., 'CSF*'                                     */
/* csfusers_grp - Group name for authorized ICSF users    */
/*      e.g., 'ICSFUGRP'                                   */
/*****
/*csfkeys_profile='IRR.DIGTCERT.** */
/*csfserv_profile='CSF*'                                     */
/*csfusers_grp=''                                           */
/* Change 8 */
/*****
/* Question 3 - Back up your private key?                  */
/*                                                         */
/* The exec will prompt you to enter a pass phrase to encrypt a */
/* backup copy of your CA's certificate and private key.      */
/* Caution, the text you enter at the prompt WILL be displayed */
/* at the terminal. Backup is highly recommended. If you do not*/
/* wish to back up your CA's certificate and private key to a */
/* pass phrase encrypted data set, set key_backup=0. The back up*/
/* may be done later if the key is not stored in ICSF.        */
/*                                                         */
/* Note, back up is not performed if the CA certificate was not */
/* created by this exec                                     */
/* *****/
key_backup=1
/*****
/* Question 4 - Set up z/OS UNIX level security?          */
/*                                                         */
/* z/OS UNIX may be set up to operate with a higher level of */
/* security than traditional UNIX. While we recommend this, it is*/
/* difficult to set up. You may want to defer this until later. */
/*                                                         */
/* If you don't want to set up UNIX security now, leave      */
/* unix_sec=2.                                               */
/*                                                         */
/* If you already have UNIX level security established and wish */
/* to continue it, set unix_sec=1.                          */
/*                                                         */
/* If you don't have UNIX level security established and wish */
/* to establish it now, set unix_sec=2. Note additional manual */
/* configuration probably will be required. This can be done */
/* by adding, removing, updating members of the two stem */
/* variables below. The pgmcntl_dsn stem contains the data set */
/* names of load libraries that need program control. The */
/* bpx_userid stem contains the user IDs of your server daemons.*/
/* (These need access to BPX.SERVER and BPX.DAEMON in the */
/* FACILITY class.) Again, you can defer this until later by */
/* leaving unix_sec=0                                       */
/*****
      unix_sec=2
      pgmcntl_dsn.0=9 /* Number of program controlled data sets below */

```

```

pgmcnt1_dsn.1="'CEE.SCEERUN'"
pgmcnt1_dsn.2="'CBC.SCLBDLL'"
pgmcnt1_dsn.3="'GLD.SGLDLNK'"
pgmcnt1_dsn.4="'GSK.SGSKLOAD'"
pgmcnt1_dsn.5="'SYS1.CSSLIB'"
pgmcnt1_dsn.6="'TCPIP.SEZALINK'"
pgmcnt1_dsn.7="'SYS1.LINKLIB'"
pgmcnt1_dsn.8="'CSF.SCSFMODE'"
pgmcnt1_dsn.9="'CSF.SCSFMODE1'"
bpx_userid.0=2 /* Number of additional bpx server ids below */
bpx_userid.1="OMVSKERN"
bpx_userid.2="pkistu"

/* Change 10 */
/*-----*/
/* Part 3 - Things you can change */
/*-----*/
/*****
/* This exec will record results to a log data set if desired. */
/* the name of the data set is specified below. If you do not */
/* want log data set recording, set log_dsn="" (Not recommended)*/
*****/
log_dsn="ANTOFF.IKYSETUP.LOG" /* Under your ID */

/* Change 11 */
/*****
/* Note.IKYCVSAM, the sample JCL to create VSAM datasets and */
/* pkiserv.conf expect the object store and ICL datasets to */
/* have PKISRV as their high level qualifier. */
/* Changing either "daemon" or "vsamhlq" will */
/* require making the same change to IKYCVSAM and pkiserv.conf */
*****/
vsamhlq="pki" /* HLQ for VSAM data sets. Same as daemon ID */

/* Change 12 */
web_label = "SSL CERT" /* Label for web server cert */

/* Change 13 */
ca_expires ="2010/01/01" /* date the CA certificate for +
should expire */
web_expires ="2010/01/01" /* date the certificate for
web server SSL should
expire */
ca_ring="CARING" /* keyring name for PKI Srvs */

/* Change 14 */
/*****
/* Data set to contain the backup copy of the CA certificate */
/* and private key. (pass phrase encrypted PKCS#12 format) */
*****/

```

```

/*****
    backup_dsn = "'PKI.CAPKI1.BACKUP.P12BIN'"

/* Change 15 */
/*****
/* Data set to contain the exported copy of the CA certificate */
/* (DER encoded). This is to be OPUT to an HFS file later to */
/* enable easy downloading by clients. */
/*****
    export_dsn = "'PKI.CAPKI1.DERBIN'"
/* Change 16 */
/*****
/* This EXEC expects the web server to be set up. If this is */
/* not the case, please refer to: */
/* z/OS HTTP Server Planning, Installing and Using. */
/* If the user ID assigned to the IBM HTTP Server Daemon is not */
/* WEBSRV, please update the assignment below. */
/*****
    webserver="WEBSTU"

/* Change 17 */
/*-----*/
/* End of configurable section */
/*-----*/
parse upper arg "RUN(" runopt ")"
if runopt = '' then
    runopt="NO"
if runopt ~= "YES" & runopt ~= "PROMPT" & runopt ~= "NO" then do
    say "syntax ex 'data-set-name(IKYSETUP)' 'run(yes | no | prompt)'"
    return 8
end
if runopt ~= "YES" & runopt ~= "PROMPT" then
    runopt="NO"
say 'IKYSETUP EXEC invoked ...'
return_code= '0'
max_return_code= '0'
logdata.0=0
if log_dsn ~= "" then do
    say "Allocating log data set" log_dsn "..."
    x = OUTTRAP(MSGS.)
    "FREE FI(IKYLOGDD)"
    "FREE DA("||log_dsn||")"
    "DELETE" log_dsn
    x = OUTTRAP('OFF')
    "ALLOCATE DA("||log_dsn||") FILE(IKYLOGDD) RECFM(V B)" ,
    " LRECL(256) DSORG(PS) BLKSIZE(2560) SP(1,1) TRACKS "
    al_rc= rc
    IF al_rc ~= 0 THEN
        do

```

```

say 'Allocation of log data set failed.'
    return 8
end
end
call logsay "RUN("runopt") requested"
if runopt="NO" then
    call logsay "Running in test mode. Commands are not being invoked"
    call logsay " "
    /*****
    /* Create the daemon and surrogate user IDs using RACF ADDUSER TSO*/
    /* command. Give them an OMVS segment since they will need access */
    /* to UNIX System Services.                                     */
    *****/
    call logsay "Creating users and groups ..."
    call tsoserv "ADDUSER " daemon "name('PKI Srvs Daemon')",
        " nopassword",
        " omvs(autoid)",
        " assize(256000000)",
        " threads(512))"

    if restricted_surrog=1 then
        resattr="restricted"
    else
        resattr=""
    call tsoserv "ADDUSER " surrog "nopassword",
        resattr,
        " omvs(autoid)",
        " name('PKI Srvs Surrogate')"
```

/*call tsoserv "SETROPTS EGN GENERIC(DATASET)" */

```

/* change 18; not recommended, should be done in another project */
call tsoserv "ADDSD '"vsamhlq".**' UACC(NONE)"

if pkigroup ^= "" then do
    call tsoserv "ADDGROUP " pkigroup "OMVS(AUTOID)"
    do i = 1 to pkigroup_mem.0
        call tsoserv "CONNECT" pkigroup_mem.i "GROUP("pkigroup")"
    end
end
end
/*****
* Give the administrators access to the VSAM data sets
* identified in the "ObjectStore" section of
* the pkiserv.conf file.
*****/
call logsay "Allowing administrators to access PKI databases ..."
call tsoserv "PERMIT '"vsamhlq".**' ID("pkigroup") ACCESS(CONTROL)"
call tsoserv "SETROPTS GENERIC(DATASET) REFRESH"
```

```

/*****
/* In order to create and sign digital certificates for others */
/* you need to define or import in RACF a Certificate Authority */
/* certificate and associated private key. */
/* This is done using the RACF RACDCERT GENCERT command. */
*****/
if ca_dn ^= "" then do
  call logsay "Creating the CA certificate ..."
  certcmd = "RACDCERT GENCERT CERTAUTH SUBJECTSDN("ca_dn")",
    " WITHLABEL('ca_label') NOTAFTER(DATE("ca_expires"))"
  if use_icsf=1 & key_backup=0 then
    certcmd= certcmd || " ICSF"
  call tsoserv certcmd
  if key_backup=1 then do
    /*****
    /* Export certificate and key to PKCS#12 dataset */
    *****/
    say ""
    say "Enter a passphrase to protect the key. You will need"
    say " this value later if you need to restore the key."
    say ""
    say "Attention, the value will be displayed in the screen:"
    parse pull pp
    call logsay "Backing up the CA certificate ..."
    certcmd = "RACDCERT CERTAUTH EXPORT(LABEL('ca_label'))",
      " DSN("backup_dsn") FORMAT(PKCS12DER)",
      " PASSWORD('*****')"

    call tsoserv certcmd
  end
  if use_icsf=1 & key_backup=1 then do
    /*****
    /* If ICSF was requested and key backup, reload the certificate */
    /* to get the key migrated to ICSF */
    *****/
    call logsay "Migrating the CA's private key to ICSF ..."
    certcmd = "RACDCERT CERTAUTH ADD("backup_dsn")",
      " PASSWORD('*****') ICSF"
    call tsoserv certcmd
  end
end /* ca_dn ^= "" */
/*****
/* Mark the CA certificate as HIGHTRUST so HostIdMappings */
/* are honored */
*****/
call logsay "Marking CA certificate as HIGHTRUST ..."
certcmd = "RACDCERT CERTAUTH ALTER(LABEL('ca_label')) HIGHTRUST"
call tsoserv certcmd
/*****

```



```

/* The CA certificate must be saved to a data set so that it may */
/* be OPUT to an HFS file. */
/*****
call logsay "Saving the CA certificate to a data set for OPUT ..."
certcmd = "RACDCERT CERTAUTH EXPORT(LABEL('ca_label'))",
         " DSN("export_dsn") FORMAT(CERTDER)"
call tsoserv certcmd
/*****
/* The CA certificate must be placed in a key ring so that */
/* PKI Services can access it. */
/*****
call logsay "Creating the PKI Services keyring ..."
call tsoserv "RACDCERT ADDRING("ca_ring") ID("daemon")"
call tsoserv "RACDCERT ID("daemon") CONNECT(CERTAUTH",
         " LABEL("ca_label"))",
         " RING("ca_ring") USAGE(PERSONAL) DEFAULT) "
/*****
/* Create the certificate for the webserver signed by your new CA */
/*****
if web_dn ^= "" then do
    call logsay "Creating the Webserver SSL certificate and keyring ..."
    call tsoserv "RACDCERT GENCERT ID("webserver") SIGNWITH(CERTAUTH",
         " LABEL("ca_label"))",
         " WITHLABEL("web_label") SUBJECTSDN("web_dn")",
         " NOTAFTER(DATE("web_expires"))"

/*****
/* Add the certificate to the webserver's RACF (SAF) key ring */
/*****
    call tsoserv "RACDCERT ADDRING("web_ring") ID("webserver")"
    call tsoserv "RACDCERT ID("webserver") CONNECT(ID("webserver")",
         " LABEL("web_label")) RING("web_ring") USAGE(PERSONAL) DEFAULT)"
end /* web_dn ^= "" */
/*****
/* Add the CA certificate to the webserver's RACF (SAF) key ring*/
/*****
if web_ring ^= "" then
    call tsoserv "RACDCERT ID("webserver") CONNECT(CERTAUTH",
         " LABEL("ca_label")) RING("web_ring"))"
if unix_sec = 0 then do
/*****
/* Not setting up z/OS UNIX higher security. However, the */
/* daemon does need access to one server service. So, if the */
/* daemon user ID is not uid 0, then it must be given read */
/* access to FACILITY class profile BPX.SERVER */
/*****
    if strip(daemon_uid,L,'0') ^= "" then do /* if daemon not uid 0 */
        call logsay "Giving" daemon "access to BPX.SERVER ..."
        call tsoserv "RDEFINE FACILITY BPX.SERVER"

```

```

        call tsoserv "PERMIT BPX.SERVER CLASS(FACILITY)",
            " ID("daemon") ACCESS(READ)"
    end
end
else do
    call logsay "Setting up or modifying z/OS UNIX security ..."
    F1=Help    F2=Split    F3=Exit    F5=Rfind    F7=Up        F8=Down    F
    if unix_sec = 2 then do
        /*****
        /* Set up z/OS UNIX to operate with a higher level of          */
        /* security than traditional UNIX, by defining BPX.SERVER and  */
        /* BPX.DAEMON classes.                                         */
        *****/
        call tsoserv "RDEFINE FACILITY BPX.SERVER"
        call tsoserv "RDEFINE FACILITY BPX.DAEMON"
        do i = 1 to bpx_userid.0
            call tsoserv "PERMIT BPX.SERVER CLASS(FACILITY)",
                " ID("bpx_userid.i") ACCESS(READ)"
            call tsoserv "PERMIT BPX.DAEMON CLASS(FACILITY)",
                " ID("bpx_userid.i") ACCESS(READ)"
        end
    end
    /*****
    /* To use the higher level of security, you need to establish    */
    /* RACF program control and enable the PKI Services daemon      */
    /* user ID and webserver daemon user ID to access protected     */
    /* UNIX daemon services.                                         */
    *****/
    call tsoserv "PERMIT BPX.SERVER CLASS(FACILITY) ID("daemon")",
        " ACCESS(READ)"
    call tsoserv "PERMIT BPX.DAEMON CLASS(FACILITY) ID("daemon")",
        " ACCESS(READ)"
    call tsoserv "PERMIT BPX.SERVER CLASS(FACILITY) ID("webserver")",
        " ACCESS(READ)"

    /*change 19; no need for UPDATE access */

    call tsoserv "PERMIT BPX.DAEMON CLASS(FACILITY) ID("webserver")",
        " ACCESS(READ)"
    if unix_sec = 2 then do
        /*****
        /* Set the PKI Services daemon and DLLs up for program control */
        *****/
        call tsoserv "RDEFINE PROGRAM ** UACC(NONE)"
        do i = 1 to pgmcntl_dsn.0
            call tsoserv "RALTER PROGRAM **

ADDMEM("pgmcntl_dsn.i//NOPADCHK)",
    " UACC(READ)"

```

```

        end
        call tsoserv "SETROPTS WHEN(PROGRAM)"
    end

    call tsoserv "PERMIT ** CLASS(PROGRAM)",
        " ID("surrog") ACCESS(READ)"
        call tsoserv "SETROPTS WHEN(PROGRAM) REFRESH"
end /* unix_sec -= 0 */
/* Change 20 We are using profile ** in class PROGRAM */
/*****
/* Allow the daemon to be a certificate authority */
/*****
call logsay "Allowing the PKI Services daemon to act as a CA ..."
call tsoserv "RDEFINE FACILITY IRR.DIGTCERT.GENCERT"
call tsoserv "RDEFINE FACILITY IRR.DIGTCERT.LISTRING"
call tsoserv "RDEFINE FACILITY IRR.DIGTCERT.LIST"
call tsoserv "PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY)",
    " ID("daemon") ACCESS(CONTROL)"
call tsoserv "PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY)",
    " ID("daemon") ACCESS(READ)"
call tsoserv "PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY)",
    " ID("daemon") ACCESS(READ)"
/*****
/* Allow the webserver to access its keyring */
/*****
call logsay "Allowing the Webserver to access its keyring ..."
call tsoserv "PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY)",
    " ID("webserver") ACCESS(READ)"
call tsoserv "PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY)",
    " ID("webserver") ACCESS(READ)"
/*****
/* Permit the webserver daemon User ID to switch identity to the */
/* surrogate Id */
/*****

call logsay "Allowing the Webserver to switch identity to "surrog" ..."
call tsoserv "SETROPTS CLASSACT(SURROGAT)"
call tsoserv "RDEFINE SURROGAT BPX.SRV."surrog
call tsoserv "PERMIT BPX.SRV."surrog" CLASS(SURROGAT)",
    " ID("webserver") ACCESS(READ)"
call tsoserv "SETROPTS RACLIST(SURROGAT) REFRESH"
if use_icsf then do
/*****
/* Allow the daemon authorization to use ICSF */
/*****
    call logsay "Allowing the PKI Services daemon to use ICSF ..."
    if csfkeys_profile ^= '' | csfserv_profile ^= '' then do
        call tsoserv "SETROPTS GENERIC(CSFKEYS CSFSERV)"
        call tsoserv "SETROPTS GENERIC(CSFKEYS CSFSERV) REFRESH"
    end
end

```

```

if csfkeys_profile ^= '' then do
  call tsoserv "RDEFINE CSFKEYS" csfkeys_profile "UACC(NONE)"
  call tsoserv "PERMIT" csfkeys_profile "CLASS(CSFKEYS)",
    " ID("daemon") ACCESS(READ)"
  call tsoserv "SETROPTS CLASSACT(CSFKEYS) RACLIST(CSFKEYS)"
  call tsoserv "SETROPTS RACLIST(CSFKEYS) REFRESH"
end
if csfserv_profile ^= '' then do
  call tsoserv "RDEFINE CSFSERV" csfserv_profile "UACC(NONE)"
  call tsoserv "PERMIT" csfserv_profile "CLASS(CSFSERV)",
    " ID("daemon") ACCESS(READ)"
  call tsoserv "PERMIT" csfserv_profile "CLASS(CSFSERV)",
    " ID("surrog") ACCESS(READ)"
call tsoserv "SETROPTS CLASSACT(CSFSERV) RACLIST(CSFSERV)"
  call tsoserv "SETROPTS RACLIST(CSFSERV) REFRESH"
end
if csfusers_grp ^= '' then do
  call tsoserv "CONNECT" daemon "GROUP(" csfusers_grp ")"
  call tsoserv "CONNECT" surrog "GROUP(" csfusers_grp ")"
end
end

/*****
* Tie the daemon user ID to PKI Services started procedure
*****/
call logsay "Creating the STARTED class profile for the daemon ..."
  call tsoserv "RDEFINE STARTED PKISTU.** +
  STDATA(USER("daemon"))"

/* Change 21 */
call tsoserv "SETROPTS CLASSACT(STARTED) RACLIST(STARTED)"
  call tsoserv "SETROPTS RACLIST(STARTED) REFRESH"

/*****
/* Give the surrogate user ID authority to request certificate */
/* generation functions. */
*****/
call logsay "Allowing "surrog" to request certificate functions ..."
call tsoserv "SETR GENERIC(FACILITY)"
call tsoserv "RDEFINE FACILITY IRR.RPKISERV.**"
call tsoserv "PERMIT IRR.RPKISERV.** CLASS(FACILITY) ID("surrog")",
  " ACCESS(CONTROL)"

/*****
/* The administrative functions of PKI Services are protected */
/* by the IRR.RPKISERV.PKIADMIN FACILITY class resource. */
/* The following commands give UPDATE access to the PKI */
*****/

```

```

/* services group to allow them to act on certificate */
/* requests and issued certificates. */
/*****/
call logsay "Creating the profile to protect PKI Admin functions ..."
call tsoserv "RDEFINE FACILITY IRR.RPKISERV.PKIADMIN"
call tsoserv "PERMIT IRR.RPKISERV.PKIADMIN CLASS(FACILITY)",
" ID("pkigroup") ACCESS(UPDATE)"
call tsoserv "PERMIT IRR.RPKISERV.PKIADMIN CLASS(FACILITY)",
" ID("surrog") ACCESS(NONE)"
call tsoserv "SETROPTS RACLIST(FACILITY) REFRESH"

/*****/
/* Done. Now write to the log */
/*****/
upper daemon vsamhql export_dsn
call logsay " "
call logsay "-----"
call logsay "Information needed for PKI Services UNIX set up:"
call logsay "-----"
call logsay " "
call logsay "The daemon user ID is:"
call logsay " " daemon
call logsay " "
call logsay "The VSAM high level qualifier is:"
call logsay " " vsamhql
call logsay,
"This is needed for the 'ObjectStore' section in pkiserv.conf"
call logsay " "
call logsay "The PKI Services' DER encoded certificate is in data set:"
call logsay " " export_dsn
call logsay,
"This must be OPUT to /var/pkiserv/cacert.der with the BINARY option"
call logsay " "
call logsay "The fully qualified PKI Services' SAF keyring is:"
call logsay " " daemon/"ca_ring
call logsay,
"This is needed for the 'SAF' section in pkiserv.conf"
call logsay " "
if ca_dn ^= "" then do
    call logsay "The PKI Services CA DN is:"
    call norm_dn ca_dn
    call logsay " " dn
    call logsay "The suffix must match the LDAP suffix in slapd.conf"
end
else
    call logsay "CA certificate not created by this exec"
call logsay " "
if web_dn ^= "" then do

```

```

        call logsay "The webserver's SAF keyring is:"
        call logsay " " web_ring
    call logsay,
        "This is needed for the KeyFile directive in httpd*.conf files"
        call logsay " "
        call logsay "The Webserver's DN is:"
        call norm_dn web_dn
        call logsay " " dn
        call logsay "The left most RDN must be the webserver's fully",
            "qualified domain name"
    end
else
    call logsay,
        "Webserver certificate and keyring not created. You must add the CA",
        "certificate as a 'trusted root' manually"
    call logsay " "
    if log_dsn ^= "" then do
        x = OUTTRAP(MSGS.)
        'EXECIO' logdata.0 'DISKW IKYLOGDD (FINIS STEM LOGDATA.'
        'FREE FI(IKYLOGDD)'
        x=OUTTRAP('OFF')
        say "Commands complete. Results written to log data set" log_dsn
    end
    /*****
    /* Exit
    /*****/
    say 'The IKYSETUP EXEC has completed.'
    Exit max_return_code
    /*****
    /* tsoserv - echo rc and commands and track highest rc
    /*****/
    tsoserv:
    Parse arg cmd
    return_code = 0
    skipit= 0
    if runopt = "NO" | runopt = "PROMPT" then
        call logsay cmd
    if runopt = "PROMPT" then do
        say "Run command (y/n) ?"
        parse pull ans
        if substr(ans,1,1) ^= 'Y' & substr(ans,1,1) ^= 'y' then
            skipit= 1
        end
    if skipit = 0 then
        if runopt = "YES" | runopt = "PROMPT" then do
            msg_status= MSG('ON')
            x=OUTTRAP('rac_ret.')
            Address TSO cmd
            return_code=rc

```

```

y=OUTTRAP('OFF')
call logsay 'Return code' return_code 'from->' cmd
If return_code\=0 then do
  Do j=1 to rac_ret.0
    call logsay rac_ret.j
  end
end
end
max_return_code= max(max_return_code,return_code)
return return_code
return 0
/*****
/* logsay - echo messages to the terminal and logdata stem */
*****/
logsay:
Parse arg cmd
parse var cmd leftpart " PASSWORD(' pw ') " rightpart
if pw ^= "" then
  cmd= leftpart "PASSWORD('*****') " rightpart
say cmd
k= logdata.0 + 1
logdata.k= cmd
logdata.0= k

return 0
/*****
/* norm_dn - transform the RACF dn keywords to an LDAP dn */
*****/
norm_dn:
parse arg in_dn
parse var in_dn q.1 "(" v.1 ")",
q.2 "(" v.2 ")",
q.3 "(" v.3 ")",
q.4 "(" v.4 ")",
q.5 "(" v.5 ")",
q.6 "(" v.6 ")",
q.7 "(" v.7 ")" rest

dns.= ""
do i = 1 to 7
  q= strip(q.i)
  upper q
  if q = "" then
    leave
  if q = "CN" then
    dns.1= "CN=" || v.i
  else
    if q = "T" then
      dns.2= "T=" || v.i
    else

```

```

if q = "OU" then
  dns.3= "OU=" || v.i
else
  if q = "O" then
    dns.4= "O=" || v.i
  else
    if q = "L" then
      dns.5= "L=" || v.i
    else
      if q = "SP" then
        dns.6= "ST=" || v.i
      else
        dns.7= "C=" || v.i
    end
  end
  dn= ""
do i = 1 to 7
  if dns.i ^= "" then
    if dn = "" then
      dn= dns.i
    else
      dn= dn || "," || dns.i
    end
  end
end
return 0

```

Example 5-2 shows the messages received in the LOG data set after running the exec with option RUN(YES).

Example 5-2 Output written to log data set from running IKYSETUP

```

RUN(YES) requested

Creating users and groups ...
Return code 0 from-> ADDUSER PKISRV name('PKI Srvs Daemon') nopassword omvs(uid(554) assize(256000000)
threads(512))
Return code 0 from-> ADDUSER PKISRV nopassword omvs(uid(555)) name('PKI Srvs Surrogate')
Return code 0 from-> SETROPTS EGN GENERIC(DATASET)
Return code 0 from-> ADDSD 'PKISRV.*' UACC(NONE)
Return code 0 from-> ADDGROUP PKIGRP OMVS(GID(655))
Return code 0 from-> CONNECT antoff GROUP(PKIGRP)
Return code 0 from-> CONNECT trauner GROUP(PKIGRP)
Allowing administrators to access PKI databases ...
Return code 0 from-> PERMIT 'PKISRV.*' ID(PKIGRP) ACCESS(CONTROL)
Return code 0 from-> SETROPTS GENERIC(DATASET) REFRESH
Creating the CA certificate ...
Return code 0 from-> RACDCERT GENCERT CERTAUTH SUBJECTSDN(OU('ITSO Poughkeepsie CA') O('IBM') C('US')) WITHLABEL('IBM
ITSO POK PKI1') NOTAFTER(DATE(2020/01/))
Backing up the CA certificate ...
Return code 0 from-> RACDCERT CERTAUTH EXPORT(LABEL('IBM ITSO POK PKI1')) DSN('PKISRV.antoff.KEY.BACKUP.P12BIN')
FORMAT(PKCS12DER) PASSWORD('*****')
Marking CA certificate as HIGHTRUST ...
Return code 0 from-> RACDCERT CERTAUTH ALTER(LABEL('IBM ITSO POK PKI1')) HIGHTRUST
Saving the CA certificate to a data set for OPUT ...
Return code 0 from-> RACDCERT CERTAUTH EXPORT(LABEL('IBM ITSO POK PKI1')) DSN('PKISRV.antoff.CACERT.DERBIN')
FORMAT(CERTDER)
Creating the PKI Services keyring ...
Return code 0 from-> RACDCERT ADDRING(Caring) ID(PKISRV)
Return code 0 from-> RACDCERT ID(PKISRV) CONNECT(CERTAUTH LABEL('IBM ITSO POK PKI1') RING(Caring) USAGE(PERSONAL)
DEFAULT)
Creating the Webserver SSL certificate and keyring ...

```



```

Return code 0 from-> RACDCERT GENCERT ID(WEBSRV) SIGNWITH(CERTAUTH LABEL('IBM ITSO POK PKI1')) WITHLABEL('SSL Cert')
SUBJECTSDN(CN('wtsc63oe.itso.ibm.com') O('IBM') L('Poughkeepsie') SP('New York') C('US')) NOTAFTER(DATE(2020/01/01))
Return code 0 from-> RACDCERT ADDRING(SSLring) ID(WEBSRV)
Return code 0 from-> RACDCERT ID(WEBSRV) CONNECT(ID(WEBSRV) LABEL('SSL Cert') RING(SSLring) USAGE(PERSONAL) DEFAULT)
Return code 0 from-> RACDCERT ID(WEBSRV) CONNECT(CERTAUTH LABEL('IBM ITSO POK K PKI1') RING(SSLring))
Giving PKISRVD access to BPX.SERVER ...
Return code 4 from-> RDEFINE FACILITY BPX.SERVER
BPX.SERVER ALREADY DEFINED TO CLASS FACILITY.
Return code 0 from-> PERMIT BPX.SERVER CLASS(FACILITY) ID(PKISRVD) ACCESS(READ)
Allowing the PKI Services daemon to act as a CA ...
Return code 4 from-> RDEFINE FACILITY IRR.DIGTCERT.GENCERT
IRR.DIGTCERT.GENCERT ALREADY DEFINED TO CLASS FACILITY.
Return code 4 from-> RDEFINE FACILITY IRR.DIGTCERT.LISTRING
IRR.DIGTCERT.LISTRING ALREADY DEFINED TO CLASS FACILITY.
Return code 4 from-> RDEFINE FACILITY IRR.DIGTCERT.LIST
IRR.DIGTCERT.LIST ALREADY DEFINED TO CLASS FACILITY.
Return code 0 from-> PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(PKISRVD) ACCESS(CONTROL)
Return code 0 from-> PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(PKISRVD) ACCESS(READ)
Return code 0 from-> PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(PKISRVD) ACCESS
Allowing the Webserver to access its keyring ...
Return code 0 from-> PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)
Return code 0 from-> PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)
Allowing the Webserver to switch identity to PKISERV ...
Return code 0 from-> SETROPTS CLASSACT(SURROGAT)
Return code 0 from-> RDEFINE SURROGAT BPX.SRV.PKISERV
Return code 0 from-> PERMIT BPX.SRV.PKISERV CLASS(SURROGAT) ID(WEBSRV) ACCESS(READ)
Return code 0 from-> SETROPTS RACLIST(SURROGAT) REFRESH
Creating the STARTED class profile for the daemon ...
Return code 0 from-> RDEFINE STARTED PKISRVD.** STDATA(USER(PKISRVD))
Return code 0 from-> SETROPTS CLASSACT(STARTED) RACLIST(STARTED)
Return code 0 from-> SETROPTS RACLIST(STARTED) REFRESH
Allowing PKISERV to request certificate functions ...
Return code 0 from-> SETR GENERIC(FACILITY)
Return code 0 from-> RDEFINE FACILITY IRR.RPKISERV.**
Return code 0 from-> PERMIT IRR.RPKISERV.** CLASS(FACILITY) ID(PKISERV) ACCESS(CONTROL)
Creating the profile to protect PKI Admin functions ...
Return code 0 from-> RDEFINE FACILITY IRR.RPKISERV.PKIADMIN
Return code 0 from-> PERMIT IRR.RPKISERV.PKIADMIN CLASS(FACILITY) ID(PKIGRP) ACCESS(UPDATE)
Return code 0 from-> PERMIT IRR.RPKISERV.PKIADMIN CLASS(FACILITY) ID(PKISERV) ACCESS(NONE)
Return code 0 from-> SETROPTS RACLIST(FACILITY) REFRESH

```

```

-----
Information needed for PKI Services UNIX set up:
-----

```

The daemon user ID is:
PKISRVD

The VSAM high level qualifier is:
PKISRVD
This is needed for the [ObjectStore] section in pkiserv.conf

The PKI Services' DER encoded certificate is in data set:
'PKISRVD.ANTOFF.CACERT.DERBIN'
This must be OPUT to /var/pkiserv/cacert.der with the BINARY option

The fully qualified PKI Services' SAF keyring is:
PKISRVD/CARing
This is needed for the [SAF] section in pkiserv.conf

The PKI Services CA DN is:
OU=ITSO Poughkeepsie CA,O=IBM,C=US
The suffix must match the LDAP suffix in slapd.conf

The webserver's SAF keyring is:
SSLring
This is needed for the KeyFile directive in httpd*.conf files

The Webserver's DN is:
CN=wtsc63oe.itso.ibm.com,O=IBM,L=Poughkeepsie,ST=New York,C=US
The left most RDN must be the webserver's fully qualified domain name

The RACF environment that is created by running the IKYSETUP exec is good. Yet, if you wish to avoid a string of violation messages, you must have achieved a

message-free RACF environment for all prerequisite products needed to install PKI Services.

PKI Exit

The PKI Exit provides advanced customization for additional authorization checking, validating, and changing parameters on calls to the R_PKIServ callable service (IRRSPX00) and capturing certificates for further processing. You can call this exit from the PKIServ CGIs and use its IRRSPX00 end-user preprocessing and post-processing functions, except the VERIFY function. An exit sample, coded in the C language, is included in *z/OS Security Server PKI Services Guide and Reference*, SA22-7693.

6.1 PKI Exit main routine

The main routine of the program determines which subroutine to call, based on the R_PKIServ function being called and whether this is a preprocessing or post-processing call. The individual subroutines in the program handle the following scenarios:

- | | |
|-------------------|--------------------------------------------------------------------------------------------------|
| Scenario 1 | Allow only selected users to request PKI browser certificates for authenticating to z/OS. |
| Scenario 2 | Maintain a customized certificate repository (DB2 database or file) independent of PKI Services. |
| Scenario 3 | Mandate a policy for certificate renewal only within 30 days of expiration. |

You can write your own exit to further customize your PKI Services as you see fit. For example, you may imbed SQL statements in your C code to do additional checking when users request certificates. This checking can be based on comparison between user-supplied values for mother's maiden name, birthplace, date of birth, or other criteria with values pre-stored in a DB2 table.

In this chapter we show how to install the exit and also test Scenario 1 for the exit usage.

6.2 Steps for installing and modifying the exit code sample

The C source code for the sample exit and a file containing the UNIX **make** command reside in the system-supplied directory /usr/lpp/pkiserv/samples. The steps to install and modify the exit sample after issuing the TSO OMVS command and becoming a superuser (you also may use the ISHELL command) are:

1. Copy the sample exit (pkixit.c) and makefile (Makefile.pkixit) to your first Web server directory:

```
cd /web/pki1/  
cp /usr/lpp/pkiserv/samples/pkixit.c pkixit.c  
cp /usr/lpp/pkiserv/samples/Makefile.pkixit Makefile.pkixit
```
2. Copy the source code to another file for later reference and modify the source code according to your needs:

```
cp pkixit.c pkixit.original  
oedit pkixit.c
```

3. Compile and link to produce the executable module pkiexit by issuing the following command:

```
make pkiexit
```

The resulting UNIX command creates two files: the executable module pkiexit and the object module pkiexit:

```
c89 -O -o pkiexit pkiexit.c
```

4. Set the permission bits for the module pkiexit:

```
chmod 755 pkiexit
```

5. Make the module program-controlled:

```
extattr +p pkiexit
```

Note: If you omit this step, the following messages will appear in the syslog when you restart the PKI Services to use the exit:

```
BPXP015I HFS PROGRAM /web/pki1/pkiexit IS NOT MARKED PROGRAM CONTROLLED.  
BPXP014I ENVIRONMENT MUST BE CONTROLLED FOR SERVER (BPX.SERVER)  
PROCESSING.
```

6. Edit both Web servers' environment variables files by issuing the following commands:

```
oedit /web/pki1/httpd.envvars  
oedit /web/pki1a/httpd.envvars
```

At the end of both files, add:

```
_PKISERV_EXIT=/web/pki1/pkiexit
```

7. Edit the PKI Services environment variable file:

```
oedit pkiserv.envvars
```

At the end of the file, add:

```
#  
# Pki Exit  
#  
_PKISERV_EXIT=/web/pki1/pkiexit
```

Now check the permission bits and the extended attributes by issuing:

```
ls -E pkiexit
```

The output is:

```
-rwxr-xr-x -ps- 1 HAIMO   SYS1      94208 May 19 16:15 pkiexit
```

6.3 Test for scenario 1

This scenario is for allowing only selected local z/OS users to request PKI browser certificates for authenticating to z/OS. Such a scenario might be useful to allow selected user IDs to authenticate themselves from the Internet to log on remotely to z/OS LPARs.

Additionally, this scenario provides a customized TITLE value for the subject's distinguished name based on the user's role in the organization. Permission and the user's role in the organization are indicated by the user's level of access to profiles PROJ.MEMBER and PROJ.PARTNER in RACF class FACILITY. The access values are:

NONE No access for either resource. The user is not permitted to request this type of certificate. The certificate request is denied.

READ to PROJ.MEMBER

The user is a team member and is permitted to request the certificate. TITLE value is set to Team Member. Certificate requests for team members are automatically approved. (No administrator approval is required.)

UPDATE to PROJ.MEMBER

The user is the team's leader and is permitted to request the certificate. TITLE value is set to Team Leader. A certificate request by the team leader is automatically approved. (No administrator approval is required.)

READ to PROJ.PARTNER

The user is considered to be a general partner of the team, not an active team member. The user is allowed to request certificates, but the requests require administrator approval before being issued. TITLE value is set to Team Partner.

UPDATE to PROJ.PARTNER

The user is considered to be a trusted partner of the team, not an active team member. The user is allowed to request certificates, and unlike requests of the general partner, the certificate requests are approved automatically. TITLE value is set to Team Trusted Partner.

The preprocessing exit call for the GENCERT and REQCERT functions (subroutine preProcessGenReqCertExit) handles the previously described logic. These are the steps:

1. The request values are passed into the exit through argv in field-name=fieldvalue pairs, and the subroutine looks for Template= and Userid= in the input parameters.
2. When the exit code finds a Template= value containing PKI Browser Certificate For Authenticating To z/OS, the _check_resource_auth_np() system function (refer to 3.8.59 in *z/OS C/C++ Run-Time Library Reference*, SA22-7821) examines the user ID. This determines the user's access to the preceding profiles.
 - If the user has no access to either of these resources, return code 8 is set. This causes the request to be denied.
 - Otherwise, the user's TITLE is set by imbedding the TITLE=title-value string into the certificate.

By default, administrator approval is not required for the PKI browser certificate for authenticating to z/OS.

- ▶ When the user has only READ access to PROJ.PARTNER, the function must be changed to require administrator approval. This is done by setting return code 4.
- ▶ For all other accesses, the function does not have to be changed.

We did not make any changes to the exit code. To test its functionality we created the profiles in class FACILITY:

```
RDEF FACILITY PROJ.MEMBER OWNER(PKIADM)
RDEF FACILITY PROJ.PARTNER OWNER(PKIADM)
```

We started testing with nobody on the access list of the profiles, then permitting user ID ANTOFF gradually with READ and UPDATE to each of the profiles.

Then, as user ID ANTOFF and not being on the access list of either profile, we requested **2 Year PKI Browser Certificate For Authenticating To z/OS**, as shown in Figure 6-1 on page 210.

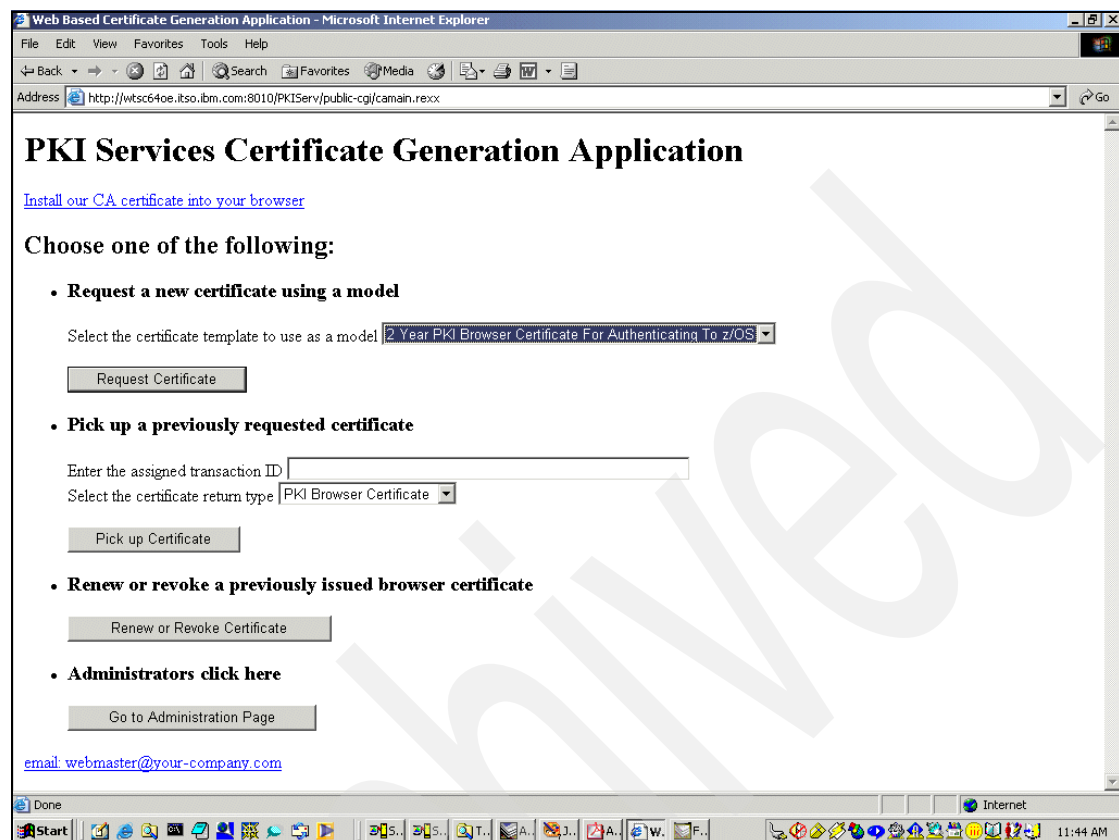


Figure 6-1 Request for 2 Year PKI Browser Certificate For Authenticating To z/OS

We clicked **Request Certificate**, which opened the window shown in Figure 6-2 on page 211. (We have entered a name and password and have selected for Cryptographic Service Provider **Microsoft Base Cryptographic Provider v1.0**, which is the recommended selection).

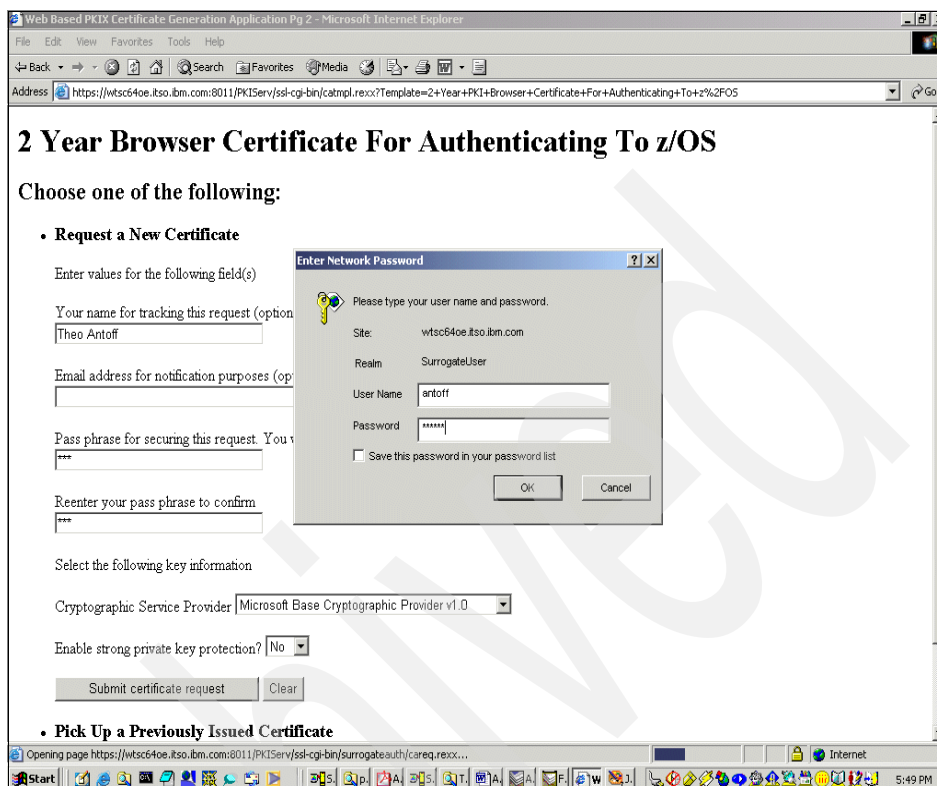


Figure 6-2 Submit certificate request

Now, after clicking **Submit certificate request**, then clicking **OK** on the pop-up authentication insert, the message in Figure 6-3 on page 212 appeared.

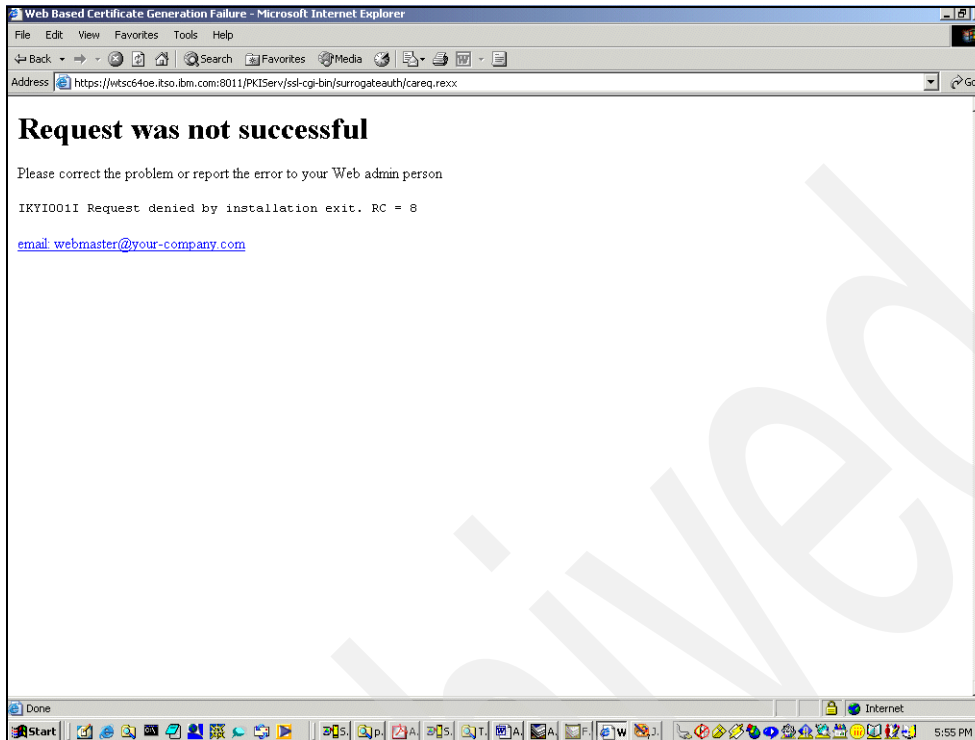


Figure 6-3 Request was not successful due to lack of access

This message shows that the user does not have access to either PROJ.MEMBER or PROJ.PARTNER profiles in RACF class FACILITY.

When we had access READ to profile PROJ.MEMBER in FACILITY and registered the issued certificate to RACF using the Selfreg application, the output from the RACDCERT LIST command appeared as Example 6-1 on page 210.

Example 6-1 Output from RACDCERT LIST (access READ to profile PROJ.MEMBER)

```
Label: LABEL00000002
Certificate ID: 2QbB1ePWxsbTwcLF0/Dw8PDw8PDy
Status: TRUST
Start Date: 2003/05/19 00:00:00
End Date: 2005/05/17 23:59:59
Serial Number:
>17<
Issuer's Name:
>OU=ITSO PSIE CA.0=IBM.C=US<
Subject's Name:
>CN=THEO ANTOFF.T=Team Member.OU=ITSO Class 1 Internet Certificate CA.<
```

```
>O=IBM.C=US<
Key Usage: HANDSHAKE
Private Key Type: None
Ring Associations:
*** No rings associated ***
```

Note: The exit added to the distinguished name a value for title T=Team Member.

When we had access UPDATE to profile PROJ.MEMBER in class FACILITY, the output from the RACDCERT LIST command was as shown in Example 6-2.

Example 6-2 Output from RACDCERT LIST (access update to profile PROJ.MEMBER)

```
Label: LABEL00000003
Certificate ID: 2QbB1ePWxsbTwcLF0/Dw8PDw8PDz
Status: TRUST
Start Date: 2003/05/19 00:00:00
End Date: 2005/05/17 23:59:59
Serial Number:
>18<
Issuer's Name:
>OU=ITSO PSIE CA.O=IBM.C=US<
Subject's Name:
>CN=THEO ANTOFF.T=Team Leader.OU=ITSO Class 1 Internet Certificate CA.<
>O=IBM.C=US<
Key Usage: HANDSHAKE
Private Key Type: None
Ring Associations:
*** No rings associated ***
```

Note: The exit added to the distinguished name a value for title T=Team Leader.

When we had access READ to profile PROJ.PARTNER in class FACILITY, we received the message shown in Figure 6-4 on page 214.

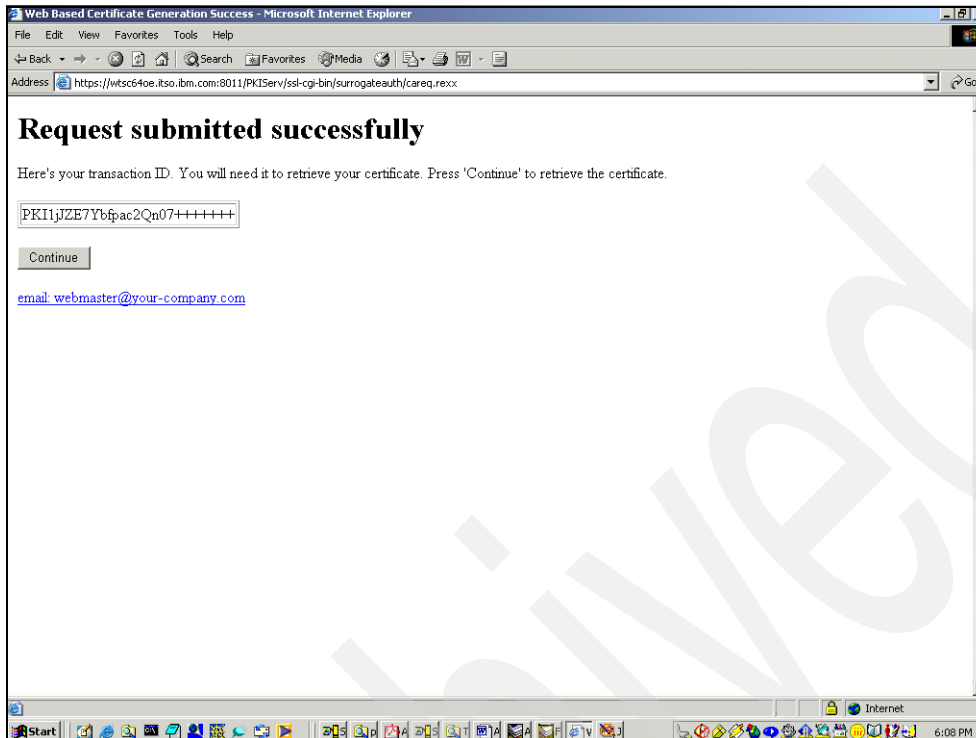


Figure 6-4 Request submitted successfully

We clicked **Continue**, which opened the window shown in Figure 6-5 on page 215.

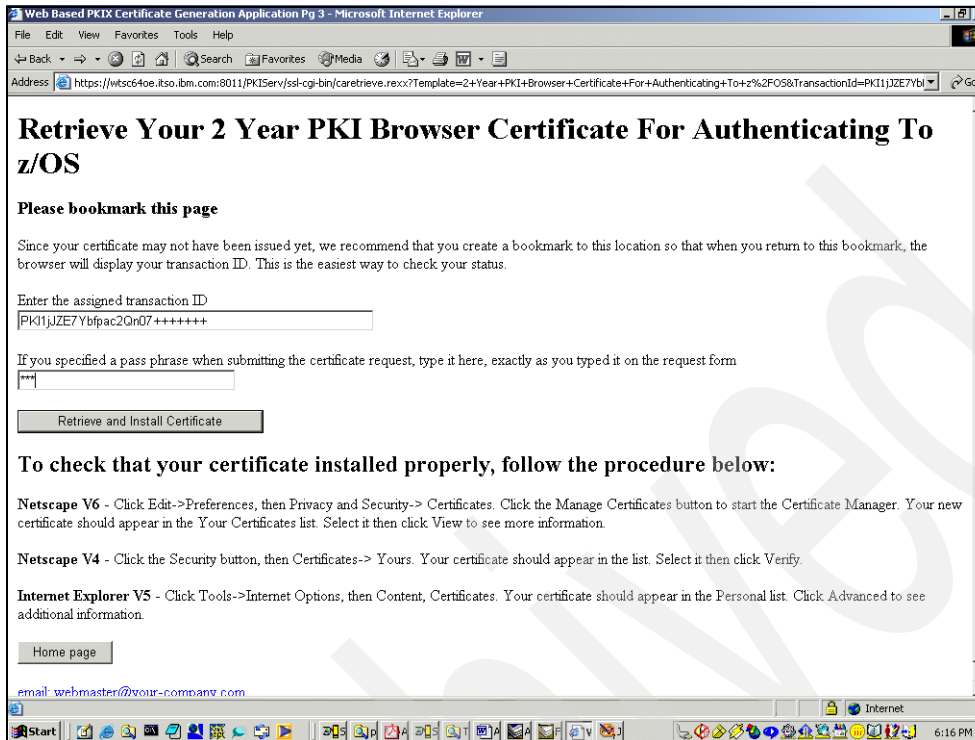


Figure 6-5 Retrieve page

After pasting our assigned transaction ID, entering our password for the certificate request, and clicking **Retrieve and Install Certificate**, we received the message shown in Figure 6-6 on page 216.

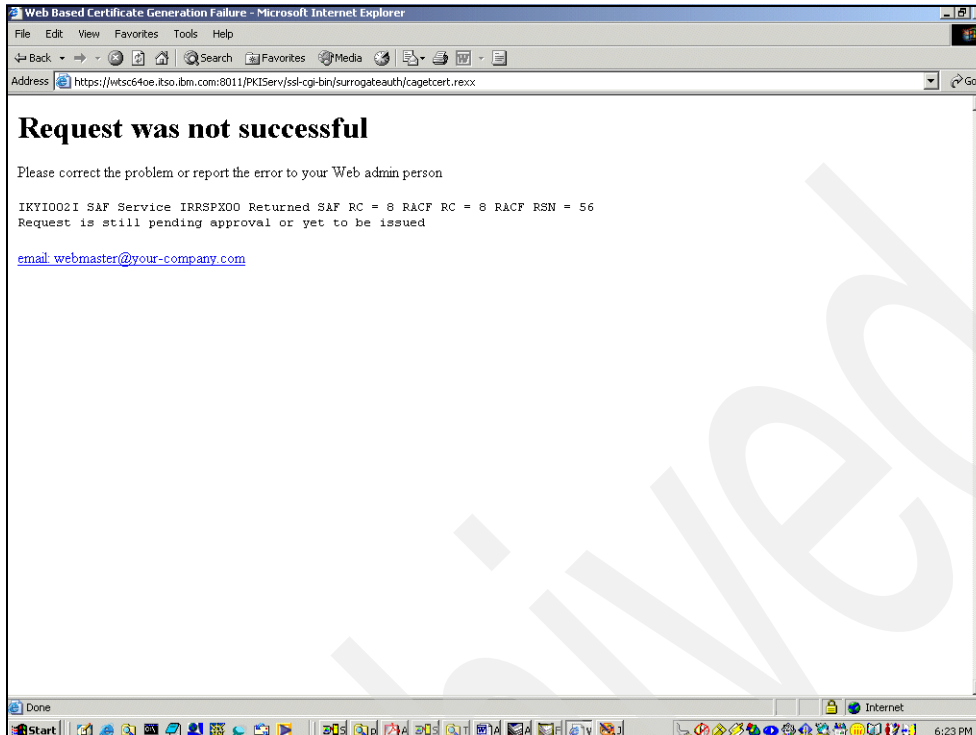


Figure 6-6 Request was not successful due to pending approval

After administrator's approval, the output of RACDCERT LIST is as shown in Example 6-3.

Example 6-3 Output from RACDCERT LIST

```

Label: LABEL00000004
Certificate ID: 2QbB1ePWxsbTwcLF0/Dw8PDw8PD0
Status: TRUST
Start Date: 2003/05/19 00:00:00
End Date: 2005/05/17 23:59:59
Serial Number:
    >1A<
Issuer's Name:
    >OU=ITSO PSIE CA.0=IBM.C=US<
Subject's Name:
    >CN=THEO ANTOFF.T=Team Partner.OU=ITSO Class 1 Internet Certificate CA<
    >.0=IBM.C=US<
Key Usage: HANDSHAKE
Private Key Type: None
  
```

Ring Associations:
*** No rings associated ***

Note: The exit added to the distinguished name a value for title T=Team Partner.

When we had access UPDATE to profile PROJ.PARTNER in class FACILITY, the output from the RACDCERT LIST is as shown in Example 6-4.

Example 6-4 Output from RACDCERT LIST (access UPDATE to PROJ.PARTNER)

Label: LABEL00000005
Certificate ID: 2QbB1ePWxsbTwcLF0/Dw8PDw8PD1
Status: TRUST
Start Date: 2003/05/19 00:00:00
End Date: 2005/05/17 23:59:59
Serial Number:
>1B<
Issuer's Name:
>OU=ITSO PSIE CA.O=IBM.C=US<
Subject's Name:
>CN=THEO ANTOFF.**T=Team Trusted Partner**.OU=ITSO Class 1 Internet Certif<
>icate CA.O=IBM.C=US<
Key Usage: HANDSHAKE
Private Key Type: None
Ring Associations:
*** No rings associated ***

Note: The exit added to the distinguished name a value for title T=Team Trusted Partner.

PKI Services and the Cryptographic Coprocessor

You will find in this chapter how PKI Services can optionally exploit the Cryptographic Coprocessor available in every S/390 G4, G5, G6, and zSeries servers. ICSF is not a prerequisite for PKI Services, but we do recommend that it be used. This chapter helps you to understand the different Cryptographic Coprocessors available in S/390, G5/G6, and zSeries servers. It also describes the new cryptographic architecture introduced in z990 and how it affects your PKI Services during migration.

This chapter includes:

- ▶ Introduction to cryptography solution on S/390 - zSeries
- ▶ Cryptographic solution on z990
- ▶ Integrated Cryptographic Services Facility
- ▶ Boosting your SSL connection with hardware encryption
- ▶ Keeping your CA signature key secure with ICSF
- ▶ Sharing PKDS in a sysplex environment

7.1 Introduction to Cryptography Solution on S/390 - zSeries

IBM offers the first CMOS Cryptographic Coprocessor in the industry as a standard feature on the S/390 G4, G5, G6 and z900 servers, and as an optional, separately priced feature on G3, Multiprise® 2000, 3000, and z800 server models, in order to meet the increasing needs for data security and integrity.

The S/390 and zSeries cryptography solution is made up of one or more cryptographic coprocessors, ICSF providing the necessary access interfaces to the cryptographic hardware and TKE for those customers requiring added security to load its master and operational keys.

- ▶ Cryptographic coprocessors are secure, high-speed hardware that performs the actual cryptographic functions.
- ▶ Integrated Cryptographic Service Facility, ICSF, provides the application programming interfaces by which applications request the cryptographic services to the cryptographic feature.
- ▶ Trusted Key Entry, TKE workstation, is an S/390 and zSeries optional-priced feature that enables you to create a logical, secure channel through which master keys and operational keys can be distributed to local and remote locations. This logical, secure channel ensures both the integrity and the privacy of the transfer channel.

Figure 7-1 on page 221 shows a high-level representation of how the cryptographic coprocessors work in S/390 and zSeries.

To Use the Cryptographic Coprocessors With z/OS

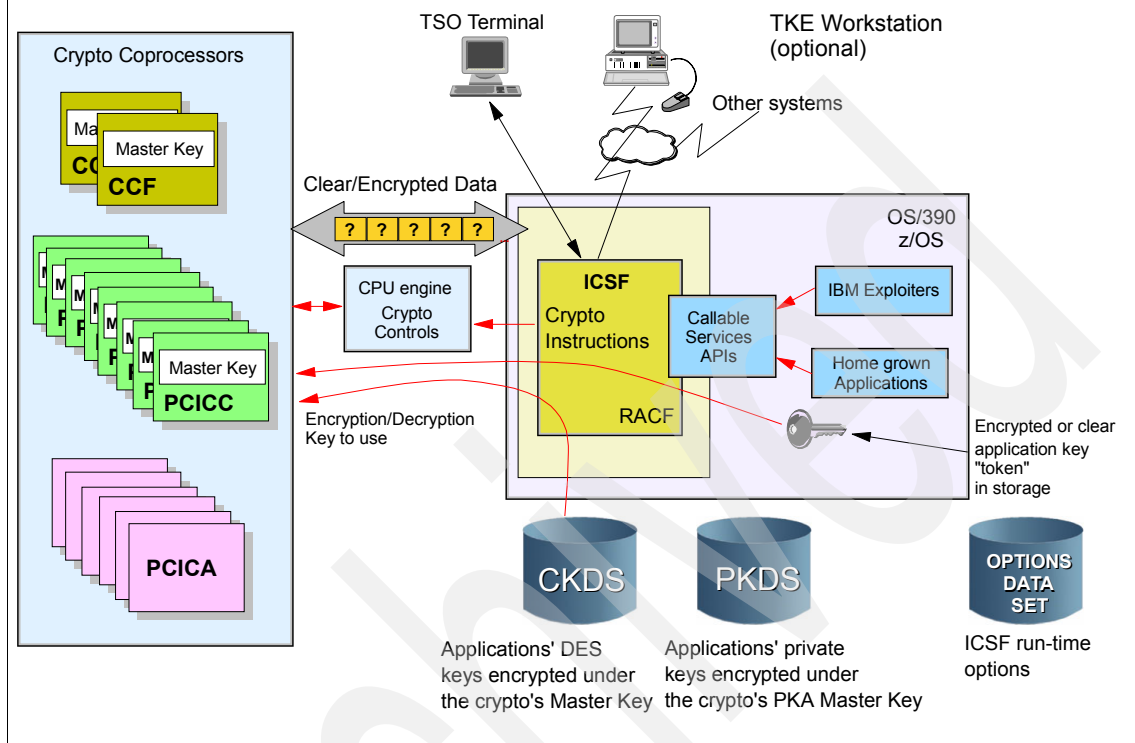


Figure 7-1 S/390 cryptographic coprocessors

7.1.1 Cryptographic Coprocessor Feature (CCF)

All S/390 G4, G5, G6, and zSeries servers have the Cryptographic Coprocessor Feature installed. All server models except the uniprocessor servers (R15, RA5, R16, 1C1, 2C1), have two Cryptographic Coprocessor Features installed. The uniprocessor models have just one CCF.

The Cryptographic Coprocessor Feature, is a high-speed extension of the central processor containing both DES and PKA cryptographic engines and a register array to store the master keys. The cryptographic coprocessor chips are protected by tamper-detection circuitry and a cryptographic battery unit.

The high-availability CMOS Cryptographic Coprocessors feature has earned Federal Information Processing Standard (FIPS) 140-1 Level 4, the highest certification for commercial security awarded by the U.S. government.

These cryptographic coprocessors are shipped disabled. An enablement diskette is required to enable them. The enablement diskette is linked to each CCF through a Crypto Module ID (CMID). The enablement diskette sets the CCF Configuration Control Code (CCC). The CCC bits determine which algorithms and key lengths are enabled to CCF.

The Integrated Cryptographic Service Facility uses a hierarchical key management approach. A master key protects all keys that are active on your system. Keys can be either clear or encrypted. A clear key is the base value of a key and is not encrypted under another key. To create an encrypted key, either a master key or a transport key is used to encrypt the base value of the key.

Master keys are the highest key level in the system. They are stored unencrypted and always remain in a secure area in the cryptographic hardware. Three types of master keys protect keys that are used with the zSeries and S/390 cryptographic feature: DES Master Key, PKA Key Management Master Key, and PKA Signature Master Key.

DES Master Key

The DES Master Key is a double-length (128-bit) key that is used to protect DES keys that reside in CKDS.

PKA Key Management Master Key

The PKA Key Management Master Key (KMMK) is a triple-length (192-bit) key. The KMMK protects PKA keys store in PKDS that are used in both the digital signature services and in the DES data key distribution functions.

PKA Signature Master Key

The PKA Signature Master Key (SMK) is a triple-length (192-bit) key. The SMK protects PKA keys that are used only in digital signature services.

Attention: If you do not have a PCICC, you should not change the PKA Master Keys because it makes all internal tokens in the current PKDS unusable. If you are storing your CA private key in a PKDS, you will lose it.

When changing PKA Master Keys it is necessary to reencipher and activate the PKDS in order to use it with the new PKA Master Key. This requires z/OS V1R2 and a PCICC on your system.

For CCF, each master key type has its own master key register. There is a fourth master key register for changing DES Master Key without interrupting the services exploiting it. This fourth register is called auxiliary master key register.

In ICSF, a domain is a set of master key registers made up of DES master key register, auxiliary register, PKA KMMK register, and PKA SMK register. There are 16 domains in each CCF. In S/390 and zSeries servers, you can define up to 15 logical partitions for each CPC. Each domain is associated to only one LPAR, so you can have a distinct master key for each one of your LPARs. Domain 0 is the default domain used in basic mode (non-LPAR mode).

Checking whether CCF is available

It is possible to check whether CCF is available by issuing the **D M=CPU** command.

Example 7-1 D M=CPU command

```
D M=CPU
IEE174I 10.46.31 DISPLAY M 127
PROCESSOR STATUS
ID  CPU  CR              SERIAL
0   +   -              090ECB2064
1   +   -              190ECB2064
2   -   .
3   -   .
4   -   .
5   -   .
6   -   .
7   N
8   N
9   N
A   N
B   N

CPC ND = 002064.1C7.IBM.02.000000010ECB
CPC SI = 2064.1C7.IBM.02.0000000000010ECB
CPC ID = 00

+ ONLINE   - OFFLINE   . DOES NOT EXIST   W WLM-MANAGED
N NOT AVAILABLE

CR          CRYPTO FACILITY
CPC ND      CENTRAL PROCESSING COMPLEX NODE DESCRIPTOR
CPC SI      SYSTEM INFORMATION FROM STSI INSTRUCTION
CPC ID      CENTRAL PROCESSING COMPLEX IDENTIFIER
```

If CCF is not defined to this LPAR, the Cryptographic Coprocessor does not appear in message IEE174I. According to the message above, both CCFs (CR) are offline.

Attention: There is no CONFIG CPU ONLINE command to bring CCFs online. Just after customizing and starting ICSF, both CCFs are brought online automatically.

To read how to customize and start ICSF properly, refer to *zSeries Crypto Guide Update*, SG24-6870.

7.1.2 PCI Cryptographic Coprocessor (PCICC)

PCI Cryptographic Coprocessor (PCICC) is an optional, separately priced feature that adds additional cryptographic function and cryptographic performance to S/390 G5, G6, and zSeries servers. Support for PCICC is provided in OS/390 V2R9 by new ICSF functions.

Each PCICC feature comprises a 4758-technology-based cryptographic coprocessor card embedded in an adapter package for installing within the I/O slots of G5, G6, or zSeries cages. The IBM 4758 Model 2 PCI Cryptographic Coprocessor is a programmable PCI card containing a 486-compatible microprocessor, custom hardware to perform DES and public key cryptographic algorithms, a hardware random number generator, and a master key register array. It also has protective shields, sensors, and control circuitry to protect against a variety of attacks against the system.

Note: On S/390 G5 and G6 servers, each PCI Cryptographic Coprocessor feature contains one cryptographic coprocessor card embedded in the adapter package, and one CHPID number is assigned. On zSeries servers, each PCICC feature contains two cryptographic coprocessor cards embedded in the adapter package, and two CHPID numbers are assigned.

The PCI Cryptographic Coprocessors feature has earned, like CCF, the Federal Information Processing Standard (FIPS) 140-1 Level 4, the highest certification for commercial security awarded by the U.S. government.

The system sees the PCICC as an Adjunct Processor (AP); that is, a coprocessor reachable through the Self-Timed Interface (STI) cable. Each PCICC card installed in the system has assigned a two-digit AP number or ID as an index used by the system LIC and ICSF.

The system also establishes the correspondence between the AP number assigned to a card and the PCICC card serial number. There is no explicit declaration of the PCICC card to ICSF. OS/390 V2.9 and later notifies ICSF of the physical presence of a PCICC card via an ENF (Event Notification Facility) signal.

As for CCF, one enablement diskette for the PCICCs installed in the system is required. The PCICC Functions Control Vector (FCV) diskette is customized to the machine's serial number, and it determines which algorithms and key lengths are enabled for all PCICCs installed in that machine.

The PCI Cryptographic Coprocessor feature coexists and augments CMOS CCF functions. ICSF transparently routes application requests for cryptographic services to one of the integrated cryptographic engines, either a CCF or a PCICC, depending on key types specified in the request or requested cryptographic function.

In order for these two types of cryptographic coprocessors to work together, it is necessary to install the same master key values for each coprocessor. The PCICC will not allow certain *weak* keys as master keys. The list of weak keys is documented in *IBM 4758 PCI Cryptographic Coprocessor CCA Basic Services Reference and Guide Version 2.40 for the IBM 4758 Models 002 and 023* under the Master_Key_Process verb. If you have an existing CCF installed with a weak master key, you cannot install that master key in the PCICC. You must change the CCF master keys and load those same master keys in the PCICCs.

User Defined Extensions (UDX) is the facility offered by the S/390 and zSeries PCICC, when running under control of ICSF on z/OS V1R2 or above, this facility enables users to design and implement their own cryptographic services to be executed within the secure boundary of the PCICC. This provides maximum flexibility to the PCICC user, along with all the protection that the card can offer.

If PCICC is installed, ICSF is able to generate RSA keys using the PKA Key Generate service. The RSA key format can be the 1024 Modulus Exponent form or the Chinese Remainder form (modulus bit lengths supported are 512 to 2048). RACF can take advantage of this function, as you can see in 7.5.1, "RACF taking advantage of ICSF" on page 235.

PCICC also supports retained keys, which are RSA keys generated within (and that never leave) the secure boundary of the PCICC. Only the domain that created the retained key can access it. This capability is a requirement to be a SET Certificate Authority. The public key and the key name for the private key are stored in the ICSF public key data set (PKDS), but the value of a retained private key never appears in any form outside the PCI card.

Note: Up to eight PCICC features may be ordered for a G5, G6, or zSeries server.

ICSF uses two types of master keys to protect keys that are used with the PCI Cryptographic Coprocessor: the Symmetric-Keys Master Key and the Asymmetric-Keys Master Key. PCICC also has the domain concept with separate

registers for the old, new, and current master key. Each PCICC has 16 domains that match the domain index in CCF.

Symmetric-Keys Master Key

The Symmetric-Keys Master Key (SYM-MK) key is a double-length (128-bit) key that is used to protect DES keys used on the PCI Cryptographic Coprocessor. SYM-MK is actually a triple-length (192-bit) master key that ICSF enforces to be equivalent to a double-length (128-bit) master key. This key must have the same value as the DES master key on CCF.

Asymmetric-Keys Master Key

The Asymmetric-Keys Master Key (ASYM-MK) is a triple-length (192-bit) key. The ASYM-MK protects PKA keys that are used on the PCI Cryptographic Coprocessor. This key must have the same value as the SMK on the zSeries and S/390 cryptographic feature.

For more about the PCI Cryptographic Coprocessor, refer to *zSeries Crypto Guide Update*, SG24-6870.

7.1.3 PCI Cryptographic Accelerator (PCICA)

The IBM PCI Cryptographic Accelerator is a non-programmable cryptographic coprocessor only available on zSeries servers and requiring z/OS 1.2 or later. This optional, priced feature is a crypto coprocessor designed specifically for exploitation by clear RSA key processing during SSL (Secure Sockets Layer) protocol. PCICA works only with clear key operations, so tamper-proof design is not required.

Each PCICA feature contains two crypto accelerator cards embedded in an adapter package, and two CHPID numbers are assigned. The adapter package is installed in I/O slots of the zSeries cages.

Note: In zSeries servers there can be a maximum of six PCICA features, and the combination of PCICC and PCICA features cannot exceed eight.

The PCICA is a very fast cryptographic processor designed to provide leading-edge performance of the complex RSA crypto operations used in SSL protocol. As many as 7,000 SSL handshake transactions per second were achieved by a 16-way z900 with four PCICA features installed. This number is limited by the number of cycles available to perform the software portion of SSL transactions.

If you have a PCICA online, toleration APAR OW49402 is required on lower levels of ICSF (OS/390 V2 R9, OS/390 V2 R10, and z/OS V1R1).

For more about PCI Cryptographic Accelerator, refer to *zSeries Crypto Guide Update*, SG24-6870.

7.1.4 Assigning coprocessors to an LPAR

The CCF is assigned to an LPAR through the Activation Profile. On the Processor Page you find the Cryptographic Coprocessors 0 and 1 are selected. By selecting one or both you are assigning one or both CCFs to be used by the LPAR being managed by this profile.

If the LPAR managed by this profile has just one central processor assigned, you will be able to select just one CCF. You can choose CCF0 or CCF1. If the LPAR has dedicated central processors and you select one or both CCFs, those you selected will be dedicated to this LPAR.

After selecting the CCF to your LPAR, a Crypto tab and a PCI Crypto tab are displayed. The PCI Crypto tab has the PCI Crypto Candidate List. A cryptographic coprocessor definition is required for logical partitions that define PCI cryptographic processors. Select which PCICCs and PCICAs are potentially accessible by this logical partition.

Also, a PCI Crypto Online List is shown. Select which PCI Cryptographic Coprocessors will be initially online to this logical partition when it is activated. It is possible to bring a PCICC/CA online (if previously defined in candidate list) through ICSF panels.

In order to make the addition of PCI Cryptographic Coprocessors concurrent in LPAR mode, logical partitions must be predefined with the appropriate PCI Cryptographic Coprocessor number selected in its candidate list. To maximize concurrent upgrade possibilities in this area, it is recommended that all logical partitions that have a cryptographic coprocessor defined also define all possible PCI Cryptographic Coprocessors as candidates for the logical partition. This is possible even if there are no PCI Cryptographic Coprocessors currently installed on the machine.

Additional customizing is required in the Crypto tab. Refer to *zSeries Crypto Guide Update*, SG24-6870.

7.2 Cryptographic solution on z990

On the z990, a new cryptographic architecture is introduced. Three types of cryptographic features are available. If our PKI solution is to store the CA signature key in a PKDS, the migration to the new z990 can cause an impact, depending on which features are available.

7.2.1 CP Assist for Cryptographic Function

Each CP in z990 has an assist processor on the chip in support of cryptography. The CP Assist for Cryptographic Function (CPACF) provides high performance hardware for clear key symmetric cryptographic functions.

The cryptographic architecture includes DES, Triple-DES data encryption and decryption, MAC message authorization, and SHA-1 hash. The Web server in our PKI solution takes advantage of this new assist during encryption and decryption of clear key operations for SSL and during hash functions.

The CPACF does not execute PKA functions. These functions are performed by PCIX Cryptographic Coprocessor (PCIXCC) and PCI Cryptographic Accelerator (PCICA) optional features.

7.2.2 PCI Extended Cryptographic Coprocessor

The optional, priced feature PCI Extended Cryptographic Coprocessor (PCIXCC) consolidates the functions previously offered by the Cryptographic Coprocessor Feature (CCF) and the PCI Cryptographic Coprocessor (PCICC). CCF and PCICC features were dropped on z990.

Note: NO CHPID numbers are assigned to PCIXCC.

The PCIXCC is a tamperproof feature with an FIPS 140-2 level compliance rating for secure cryptographic hardware. The master keys that used to be in CCF and PCICC are stored now in PCIXCC. In order to have the operational keys stored in CKDS and PKDS, the PCIXCC feature is required.

When migrating to z990 with our Certification Authority signature key stored in PKDS, we have two options:

- For security reasons, we recommend the PCIXCC when switching to the new equipment. It provides a smooth migration path and is the most secure place to store your CA signing key. For performance reasons, Web server SSL connections, not only for PKI but any Web server running in this system, benefit from this choice.

- ▶ If PCIXCC is not available in z990, the Certification Authority certificate must be deleted and added again to your RACF database without the ICSF keyword.

See Figure 7-2 for an overview of the z990 using CPACF and PCIXCC.

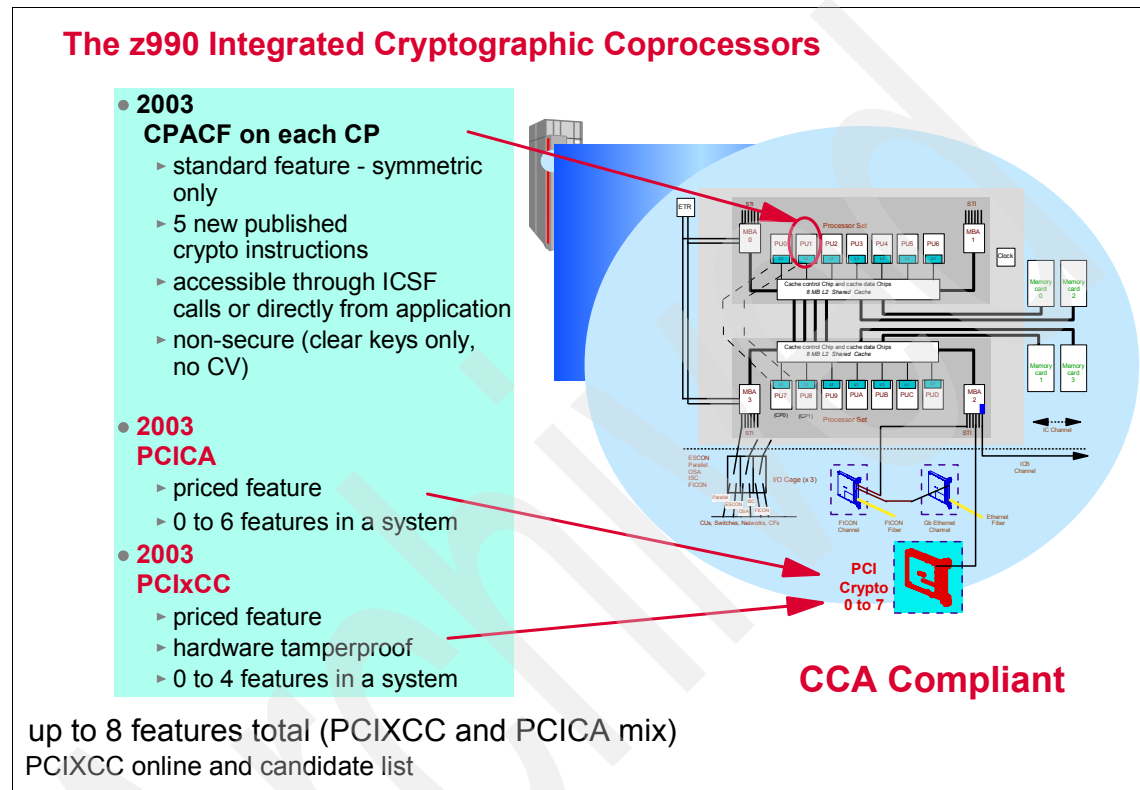


Figure 7-2 Overview of z990 with CPACF and PCIXCC

Important: Backing up the CA certificate is a very important step. Migrating your PKI Services server to z990 without PCIXCC feature requires adding the CA certificate back to the RACF database without an ICSF keyword.

The following commands were issued to add the CA certificate to RACF:

```
RACDCERT CERTAUTH DELETE(LABEL('PKISRV3 CA'))
RACDCERT CERTAUTH ADD('PKISERV3.PRIVATE.KEY.BACKUP.P12BIN')
PASSWORD('*****')
RACDCERT CERTAUTH ALTER(LABEL('PKISRV3 CA')) HIGHTRUST
```

```
RACDCERT ID(PKISRV3) CONNECT(CERTAUTH LABEL('PKISRV3 CA') RING(WEBPKI3)
USAGE(USAGE(PERSONAL) DEFAULT)
```

PCI Cryptographic Accelerator on z990

The same PCI Cryptographic Accelerator (PCICA) support on S/390 G5, G6, and zSeries servers is supported on z990. Each PCICA feature contains two cryptographic accelerator cards embedded in the adapter package.

Note: On z900 servers, NO CHPID numbers are assigned to PCICA.

7.2.3 Software requirements

The CP Assist for Cryptographic Function, PCI Extended Cryptographic Coprocessor, and PCI Cryptographic Accelerator features have specific software requirements, which are described in the table below.

Table 7-1 Minimum software requirements to support cryptographic features

Operating System	CPACF	PCIXCC	PCICA
z/OS V1R4 and later	Y ¹	Y ³	Y ¹
z/OS V1R3	Y ^{2, 4}	Y ⁴	Y ^{2, 4}
z/OS V1R2	Y ⁴	Y ⁴	Y ⁴
OS/390 V2R10	Y ⁵	Y ⁵	Y ⁵

¹ z/OS V1R4 z990 Compatibility support, available June 13, 2003

² z990 Cryptographic CP Assist Support for z/OS V1R3, available June 13, 2003

³ z990 Cryptographic support, available October 31, 2003

⁴ z990 Cryptographic support, available November 28, 2003

⁵ z990 Cryptographic support, available December 31, 2003

7.3 Integrated Cryptographic Services Facility

Integrated Cryptographic Services Facility (ICSF) is a software element of z/OS that works with the hardware cryptographic feature and the Security Server (RACF) to provide secure, high-speed cryptographic services in the z/OS environment. ICSF provides the application programming interfaces by which applications request the cryptographic services.

ICSF supports IBM Common Cryptographic Architecture (CCA). The CCA is based on the ANSI Data Encryption Algorithm (DEA). DEA is also known as the U.S.National Institute of Science and Technology Data Encryption Standard (DES) algorithm. ICSF also supports triple DES encryption for data privacy. Triple

DES uses three single-length keys to encipher and decipher the data. This results in a stronger form of cryptography than that available with single DES encipherment.

Figure 7-3 shows how ICSF works with other IBM software.

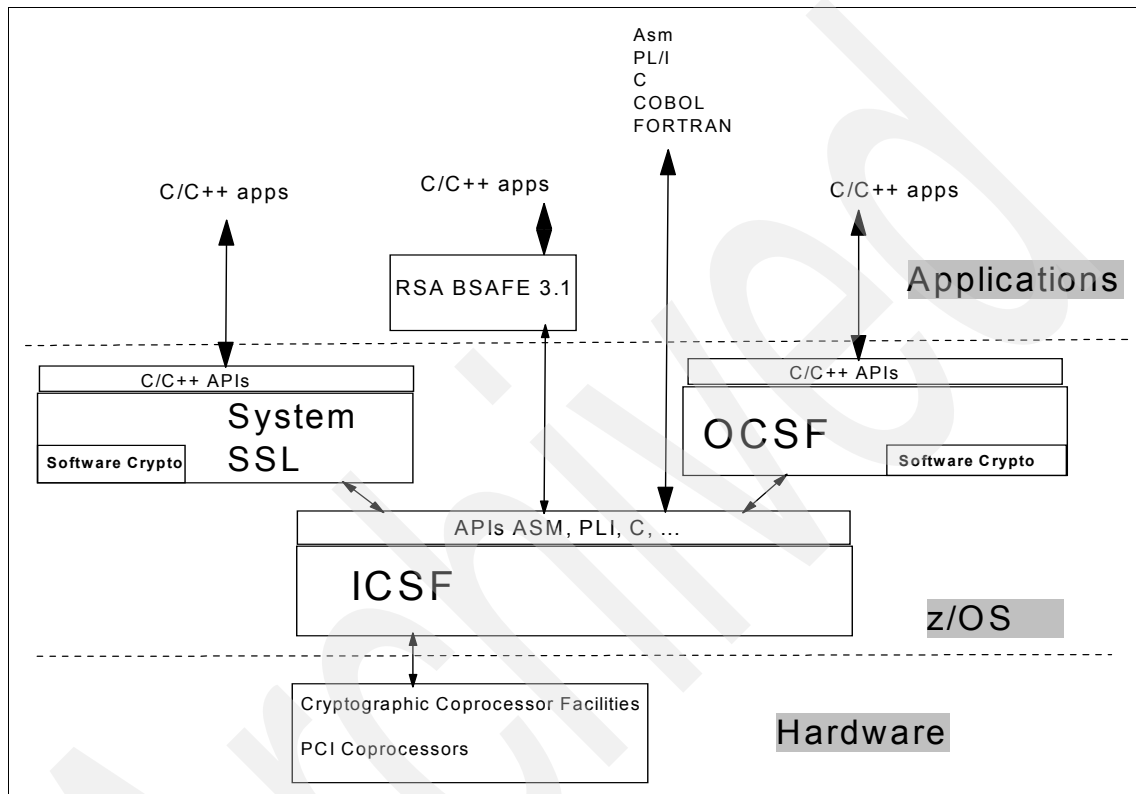


Figure 7-3 ICSF overview

Advanced Encryption Standard (AES) is also supported by ICSF. Data can be encrypted and decrypted using 128-bit, 192-bit, and 256-bit keys. CBC and ECB encryption is also supported.

For public key cryptography, ICSF supports both the Rivest-Shamir-Adelman (RSA) algorithm and the NIST Digital Signature Standard algorithm. RSA and DSS are the most widely used public key encryption algorithms. In this system, each party establishes a pair of cryptographic keys that includes a public key and a private key.

7.3.1 CKDS and PKDS

ICSF manages two VSAM data sets where cryptographic keys encrypted by master keys reside. They are the Cryptographic Key Data Set (CKDS) and the Public Key Data Set (PKDS).

Cryptographic Key Data Set

ICSF stores DES keys in a specialized data set called a Cryptographic Key Data Set (CKDS). ICSF maintains both a disk copy and an in-storage copy of the CKDS. This makes it possible to refresh the cryptographic keys without interrupting the application programs.

Public Key Data Set (PKDS)

You can store RSA and DSS public and private keys in a specialized external VSAM data set that is called a Public Key Data Set (PKDS). ICSF optionally maintains a cache of frequently used PKDS records. The size of the PKDS cache is set in the installation options data set (PKDSCACHE). It is an optional feature with a default of 64 records. If a cache is being maintained, care must be taken when deleting or changing an existing PKDS record.

7.3.2 Controlling access to ICSF resources

The control of access to ICSF cryptographic keys and services is implemented by CSFSERV and CSFKEYS classes. They must be activated and brought into storage through SETROPTS RACLIST.

CSFKEYS class controls the use of ICSF cryptographic keys. Its profiles are the key labels in CKDS and PKDS.

CSFSERV class controls the use of ICSF cryptographic services. Its profiles are the ICSF Application Programming Interfaces by which applications request the cryptographic services. The use of these classes can be seen in 2.3.5, "RACF for ICSF" on page 60

7.4 Boosting SSL connection with hardware encryption

This section describes boosting the SSL connection with hardware encryption.

7.4.1 Secure Sockets Layer (SSL)

SSL is an industry-standard protocol that the Netscape Development Corporation designed to provide a data security layer between application

protocols and TCP/IP. The SSL security protocol is widely deployed in applications on both the Internet and private intranets. SSL defines methods for data encryption, server authentication, message integrity, and client authentication for a TCP/IP connection.

SSL uses public key and symmetric techniques to protect information. SSL requires the decryption of a 48-byte SSL seed and the manipulation of this seed in the clear to produce symmetric session keys. Encrypting and decrypting this seed with public key algorithms is a very intensive CPU task.

Figure 7-4 shows an overview of how SSL uses ICSF.

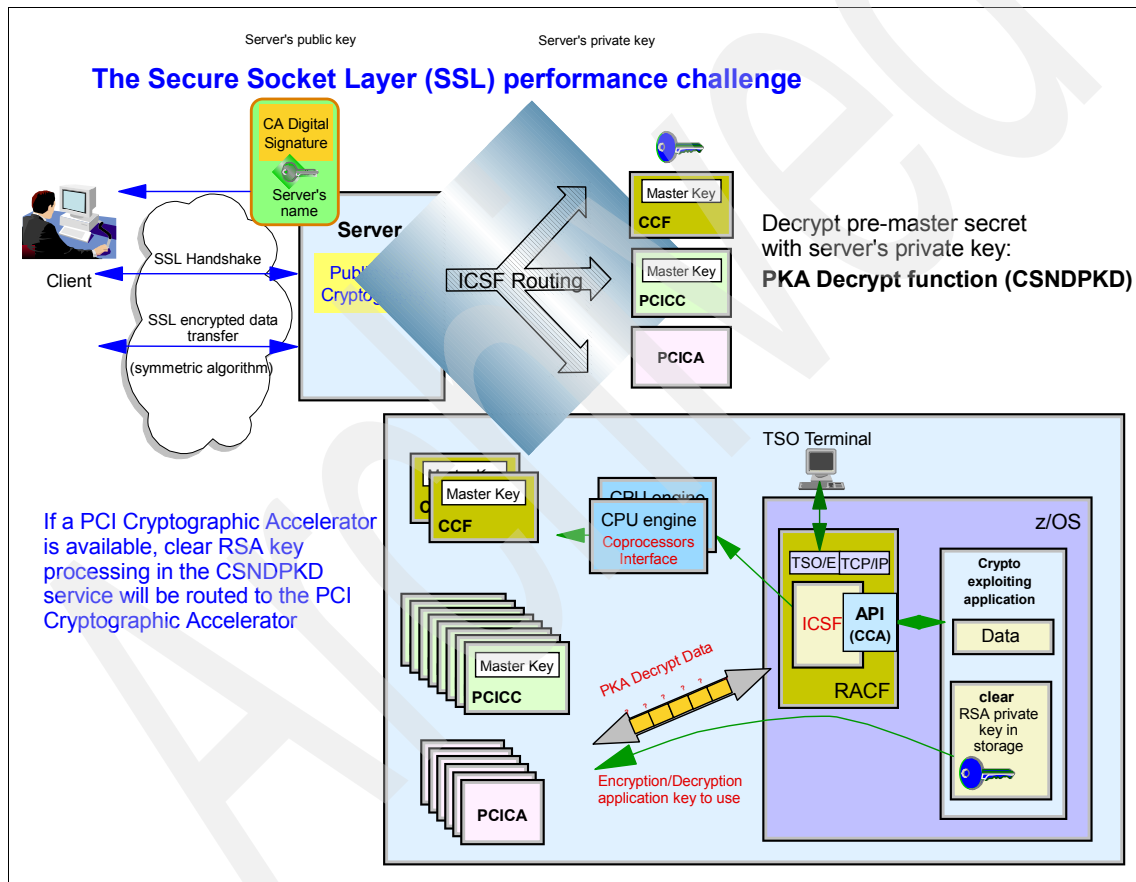


Figure 7-4 Overview of SSL

Using hardware encryption is highly recommended for improving performance of SSL sessions between the client and the server. RSA private key encryption and

decryption using the hardware cryptographic coprocessor features can take over 70 percent of the CPU cycles necessary to process the handshake.

7.4.2 IBM HTTP Server accessing the cryptographic coprocessor

There is no specific setup in the `httpd.conf` file for enabling the SSL function to access ICSF. When the IBM HTTP Server detects that ICSF is up and running during its starting process, it will start to exploit it.

The IBM HTTP server detects whether ICSF is running only during the server starting process. If IBM HTTP Server starts before ICSF, it will not take advantage of hardware encryption. Even during a server restart process, ICSF will not be checked again.

ICSF must be up and running before starting IBM HTTP Server.

CCF must be used for cryptographic hardware and, optionally, PCICC and PCICA can be used depending on the model of your processor.

7.4.3 Checking hardware encryption for Web server encryption

`GSK_SSL_HW_DETECT_MESSAGE` and `GSK_SSL_ICSF_ERROR_MESSAGE` environment variables cause System SSL to write status messages to `stderr` and to the `-vv` trace during initialization and when an ICSF error is detected.

When System SSL detects that ICSF is up and running during the IBM HTTP Server initialization, and `SSLMODE` is set to `ON`, this message is displayed:

Crypto Hardware is being used by SSL.

If ICSF is not running in the same conditions, this message is issued:

Crypto Hardware is not being used by SSL.

This support was dropped in z/OS V1R4 base code, but it was restored by APAR OA02783. The initialization status messages displayed when `GSK_SSL_HW_DETECT_MESSAGE` is set to 1 is a subset of the messages written to the SSL trace. Included are the ICSF FMID, the DES and Triple DES capabilities, and the CCF, PCICC, and PCICA availability.

Example 7-2 on page 234 shows the messages being written on z/OS V1R4 when ICSF was running when IBM HTTP Server was brought up.

Example 7-2 ICSF Messages when HTTP Server started

System SSL: ICSF FMID is HCR7708

System SSL: CCF with a valid master key is available

System SSL: DES CCF support is available
System SSL: DES3 CCF support is available
System SSL: PCI cryptographic coprocessor is available
System SSL: PCI cryptographic accelerator is not available
System SSL: Public key hardware support is available

If ICSF was not running when HTTP Server was brought up, the message in Example 7-3 is shown.

Example 7-3 Message when HTTP Server starts without ICSF running

System SSL: ICSF services are not available

The error messages displayed when GSK_SSL_ICSF_ERROR_MESSAGE is set to 1 include the name of the failing ICSF function, the return code, and the reason code.

7.5 Keeping your CA signature key secure with ICSF

Here we look at how to keep your CA signature key secure by using ICSF.

7.5.1 RACF taking advantage of ICSF

In order for PKI Services to exploit ICSF, it must be running and configured for Public Key Algorithms (PKA).

Restriction: This option should be used only if there is a PCICC available on the system and you are running z/OS V1R2 or above. These conditions enable PKDS reenciphering that avoids CA signature key loss in case PKA master keys are changed.

IKYSETUP use_icsf variable determines whether ICSF is going to be used for private key protection. By default, use_icsf variable is set to 0, indicating that ICSF is not going to be used:

```
Creating the CA certificate ...  
Return code 0 from-> RACDCERT GENCERT CERTAUTH SUBJECTSDN(CN('PKISRV3 CA')  
O('ITSO PKI') C('US')) WITHLABEL('PKISRV3 CA') NOTAFTER(DATE(2020/01/01))  
Backing up the CA certificate ...  
Return code 0 from-> RACDCERT CERTAUTH EXPORT(LABEL('PKISRV3 CA'))  
DSN('PKISERV3.PRIVATE.KEY.BACKUP.P12BIN') FORMAT(PKCS12DER)  
PASSWORD('*****')  
Migrating the CA's private key to ICSF ...
```

```
Return code 0 from-> RACDCERT CERTAUTH  
ADD('PKISERV3.PRIVATE.KEY.BACKUP.P12BIN') PASSWORD('*****') ICSF
```

If the RACDCERT GENCERT command is issued without a request-data-set-name keyword specified, a key pair is generated. If the PCICC keyword is specified in the command, the key pair is generated using the PCI Cryptographic Coprocessor. If ICSF is specified, the key pair is generated using software. In either case, the resulting private key is stored in the ICSF PKDS.

RACF can use ICSF's Public Key Data Set (PKDS) to store the PKI Services CA signing key. PKDS entries are encrypted by a Triple-DES master key stored in a register in CCF and, optionally, also in PCICC. The ICSF keyword above specifies that RACF should attempt to store the private key associated with this certificate in the ICSF PKDS. An existing certificate profile containing a non-ICSF private key is replaced by issuing the ADD keyword. If the ICSF keyword is not specified, the key is stored in the RACF database as a non-ICSF key.

If user_icsf variable was set to 0 when IKYSETUP was run, it is possible to migrate non-ICSF private keys to ICSF. It is necessary to issue the following commands to do so:

```
RACDCERT CERTAUTH ADD('PKISERV3.PRIVATE.KEY.BACKUP.P12BIN')  
PASSWORD('*****') ICSF  
RACDCERT CERTAUTH ALTER(LABEL('PKISRV3 CA')) HIGHTRUST
```

Backing up the CA certificate is a very important step. Migrating your PKI Services server to z990 without PCIXCC Feature requires adding the CA certificate back to the RACF database without the ICSF keyword.

After adding your CA certificate to RACF and storing the signing key in PKDS, you will not be allowed to back it up to a PKCS12 format file. ICSF will not bring the private key unencrypted to be exported.

CCF and PCICC are specifically designed for high security. The secure registers are not accessible through either internal code or scanning of the hardware. They are also protected by tamper-detection circuitry. Both devices are certified for Federal Information Processing Standard (FIPS) 140-1 level 4. For these reasons, we believe that PKDS is the best place to keep the PKI Services CA signing key.

The ICSF is specified for storage of private keys by using the ICSF keyword on the RACDCERT GENCERT and RACDCERT ADD commands. If the customer has PCICC already deployed, you can also use ICSF to generate private keys using the PCICC keyword on the RACDCERT GENCERT command.

When adding a certificate with a private key to be stored in ICSF, RACF creates an ICSF key label in the format

```
IRR.DIGTCERT.userid.cvtsname.ebcdic-stck-value
```

Here, *userid* is the owning user ID, *cvtsname* is the system name, as taken from CVT, and *ebcdic-stck-value* is an EBCDIC version of the current store clock value. There is no way to inform an ICSF key label when the private key is being stored, as RACFCERT ADD and RACDCERT GENCERT does not have a keyword for label.

7.6 Sharing PKDS in a sysplex environment

ICSF is supported in a sysplex environment. Both the CKDS and PKDS can be shared across systems in a sysplex.

In our PKI Services environment, only PKDS is going to be used for storing the Certification Authority signature key. That is the reason for just describing PKDS in a Parallel Sysplex environment.

The systems sharing a PKDS may be on different LPARs on the same system or different systems across multiple zSeries and S/390 processors. The only requirement for sharing the PKDS is that the same PKA Master Keys be installed on all systems sharing that PKDS. Sharing the PKDS across a sysplex is not required. Each system may have its own PKA Master Keys and its own PKDS. A sysplex may have a combination of systems that share a PKDS and individual systems with separate PKDSs.

It is highly recommended that the SMK and KMMK be the same on all systems sharing the PKDS in order to re-encipher the PKDS after a PKA master key change. PKDS reencipher requires a PCICC on your system. PKDS reencipher is not supported on CCF-only systems.

In addition, ICSF optionally maintains a cache of frequently used PKDS records. The size of the PKDS cache is set in the installation options data set. It is an optional feature, with a default of 64 records.

If a cache is being maintained, care must be taken when deleting or changing an existing PKDS record. When such an update is made on one system, that change is not automatically reflected in the cache of other systems. To ensure the integrity of the cache after PKDS updates, the PKDS cache on other systems should be refreshed.

Note: Adding PKDS records does not require refreshing of the PKDS cache.

LDAP enhancements for availability

This chapter looks at optional enhancements that can be made to LDAP to improve availability. We discuss replication and go through how to set this up, step by step. We show how to set up your primary and replica LDAP servers.

8.1 Optional LDAP enhancements for availability

When PKI Services is implemented with all of its features and functions, the LDAP server becomes a critical component to the infrastructure. Therefore, it should be configured to be available at all times. There are a couple of methods to improve the availability of the z/OS LDAP server. One method is to set up a replica server and replicate all changes from one LDAP directory to another. Another method is to set up multiple LDAP servers using the same LDAP directory in multiserver mode. This second method has some advantages but requires that systems be in a Parallel Sysplex and DB2 be in data sharing mode.

8.1.1 Redundancy

With PKI Services, multiple LDAP servers can be identified with the configuration file. If something should happen so that the first LDAP server is not available, then the second LDAP server can be used. This was set up and tested using the following methods.

Replication of the LDAP server

The first method used a replica LDAP server. In this example, the primary LDAP server was running on one LPAR in single-server mode; that is, the LDAP server had its own DB2 database, which happened to be in a separate DB2 subsystem. The secondary or replica LDAP server was running in a different LPAR in single-server mode. These were set up following the instructions below. The advantage in this setup was that every time PKI Services revoked a certificate, the change was marked in both LDAP directories. When everything was working correctly, all changes made to the primary LDAP server were replicated to the secondary LDAP server. When the connection was lost to one of the LDAP servers the update was made to the remaining LDAP server, and when the LDAP server or PKI Services was restarted, then the change was replicated to the other LDAP server. The disadvantage in this setup is that multiple writes are going to the LDAP servers to ensure that they are synchronized with each other, so there is a performance consideration. Also, there is the possibility that the LDAP directories might get out of sync with each other and have to be rebuilt, so there are monitoring and maintenance considerations.

Example 8-1 on page 241 shows an example of replication.

Example using referrals and replication

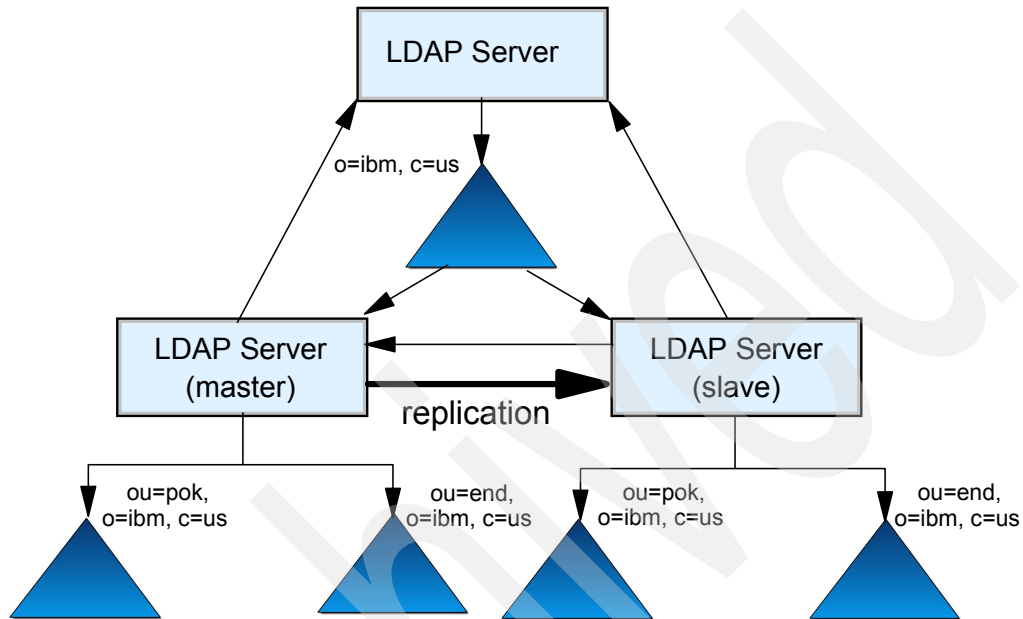


Figure 8-1 Example of replication

Setting up the primary LDAP server

Set up the LDAP server as described 3.4, “Setting up the LDAP server for PKI” on page 116. When the LDAP server is running, the schema has been loaded, the suffix and administrator have been defined, and all of this has been tested, the following steps can be used. Example 8-1 shows a configuration file for the primary LDAP server. There were no changes made to it because of replication.

Example 8-1 Configuration file for primary LDAP

```
listen ldap://wtsc63.itso.ibm.com:4389
listen ldaps://wtsc63.itso.ibm.com:4636
maxConnections 60
adminDN "cn=Admin"
sslAuth serverClientAuth
sslCipherspecs ALL
sslKeyRingFile RACF_Keyring_for_ITSOLDP
# -----
database tdbm GLDBTDBM
```

```
suffix "c=us"  
servername DB2H  
dbuserid ITSOLDP  
databasename REPM  
dsnaoini JJONES.LDAPOUT(DSNAOINI)
```

The first decision to make is whether to use a special administrator ID for this replication function. This is probably a good idea so that you have more monitoring and accessing capabilities over the different identities, but it is not a requirement. In our case, a PKI administrator identity was defined. To do this, the following UNIX command was issued from the OMVS environment:

```
ldapadd -h wtsc63.itso.ibm.com -p 4389 -D cn=Admin -w secret -f  
/u/jjones/ldaprep/defrep1dn.ldif
```

Here, `defrep1dn.ldif` is the file with the PKI administrator definition in LDIF format. It appears as in Example 8-2.

Example 8-2 defrep1dn.ldif file

```
dn: cn=PKIrepAdmin, c=us  
objectclass: top  
objectclass: person  
objectclass: organizationalPerson  
cn: PKIrepAdmin  
sn: PKIrepAdmin  
userPassword: secret
```

The only other change that has to be made to the primary LDAP server is to identify the replica LDAP server. To do this, issue the following UNIX command from the OMVS environment:

```
ldapadd -h wtsc63.itso.ibm.com -p 4389 -D cn=Admin -w secret -f  
/u/jjones/ldaprep/defrep.ldif
```

In this command, `defrep.ldif` is the file with the replica LDAP server definition in LDIF format. It appears as in Example 8-3.

Example 8-3 defrep.ldif file

```
dn: cn=PKIreplica,c=us  
objectclass: replicaObject  
cn: PKIreplica  
replicaHost: wtsc64.itso.ibm.com  
replicaPort: 5389  
replicaBindDN: cn=PKIrepAdmin,c=us  
replicaCredentials: secret
```


description: Secondary LDAP server for PKI Services
replicaUseSSL: False

The only other thing that we did was to add the new administrator. This required making changes to the appropriate ACL so that the new administrator would have enough authority to update the required LDAP subtree. In our case, the LDAP servers were only being used for PKI Services, so the ACL of the root was changed to add the new administrator. If the LDAP directory is being used by more than one service, then this might have to change. To change the ACL, the UNIX command running from the OMVS environment would look similar to:

```
ldapmodify -h wtsc63.itso.ibm.com -p 4389 -D cn=Admin -w secret -f  
modrootacl.ldif
```

The LDIF file (modrootacl.ldif) contains the information shown in Example 8-4.

Example 8-4 modrootacl.ldif

```
dn: c=us  
changetype: modify  
add: x  
aclentry:  
cn=pkirepadmin,c=us:normal:wrsc:sensitive:scwr:critical:rwsc:system:rwsc:restric  
ted:wrsc:object:ad  
-
```

Setting up the replica LDAP server

To build the replica LDAP server, follow the steps described in 3.4, “Setting up the LDAP server for PKI” on page 116. When the replica LDAP server is running, the schema is loaded (use the exact same schema definitions that were used to build the primary LDAP server), define the exact same administrators and suffix that were defined above, and modify the ACL exactly as was done in the original LDAP server setup with the minor changes to the port and IP address.

This synchronizes the LDAP directories if the directories are newly created databases. If the LDAP server had been used previously or the above process is too complex, the other method of synchronizing the database is to unload the primary directory using the TDBM utilities, then to reload the replica directory using the TDBM utilities. This is described in *z/OS Security Server LDAP Server Administration and Use*, SC24-5923. These are the same utilities that have to be used if the LDAP directories get out of synch.

The last thing that must be done within the replica LDAP server is to update the configuration file to point the replica LDAP server to the primary LDAP server. The configuration file for the replica LDAP server looks similar to Example 8-5 on page 244.

Example 8-5 Replica server configuration file

```
listen ldap://wtsc64.itso.ibm.com:5389
listen ldaps://wtsc64.itso.ibm.com:5636
maxConnections 60
adminDN "cn=Admin"
adminPW "secret"
sslAuth serverClientAuth
sslCipherspecs ALL
sslKeyRingFile RACF_Keyring_for_ITSOLDP
logfile //DD: LOGOUT
# -----
#database sdbm GLDBSDBM
#suffix "racfdb=local"
# -----
database tdbm GLDBTDBM
suffix "c=us"
masterServer ldap://wtsc63.itso.ibm.com:4389
masterServerDN cn=PKIrepAdmin,c=us
masterServerPW secret
servername DB2R
dbuserid ITSOLDP
databasename REPS
dsnaoini JJONES.LDAPOUT(DSNAOIN2)
#attroverflowsizes 255
pwEncryption none
extendedgroupsearching on
```

Starting the servers

Now start the LDAP servers and test them by adding an entry to one of the LDAP servers. This should update the data within both directories. We show an example of this in the next few pages. First, we show the LDAP servers before both our primary LDAPM and replica LDAPS using the LDAP browser before adding the new entry. Then we show them after the entry has been added. Figure 8-2 on page 245 shows our LDAP browser with entries for both LDAPM and LDAPS. First, we connect to LDAPM, and Figure 8-3 on page 245 shows LDAPM before the entry is added. Then we disconnect from LDAPM and connect to LDAPS. Figure 8-4 on page 246 shows the entries in LDAPS before the new entry is added.

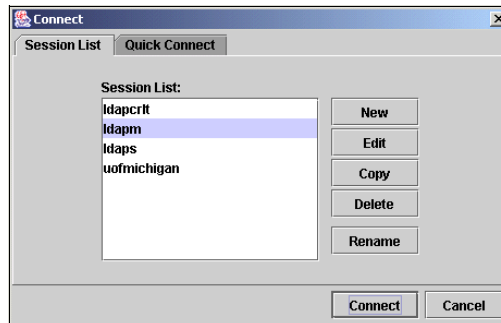


Figure 8-2 LDAP browser

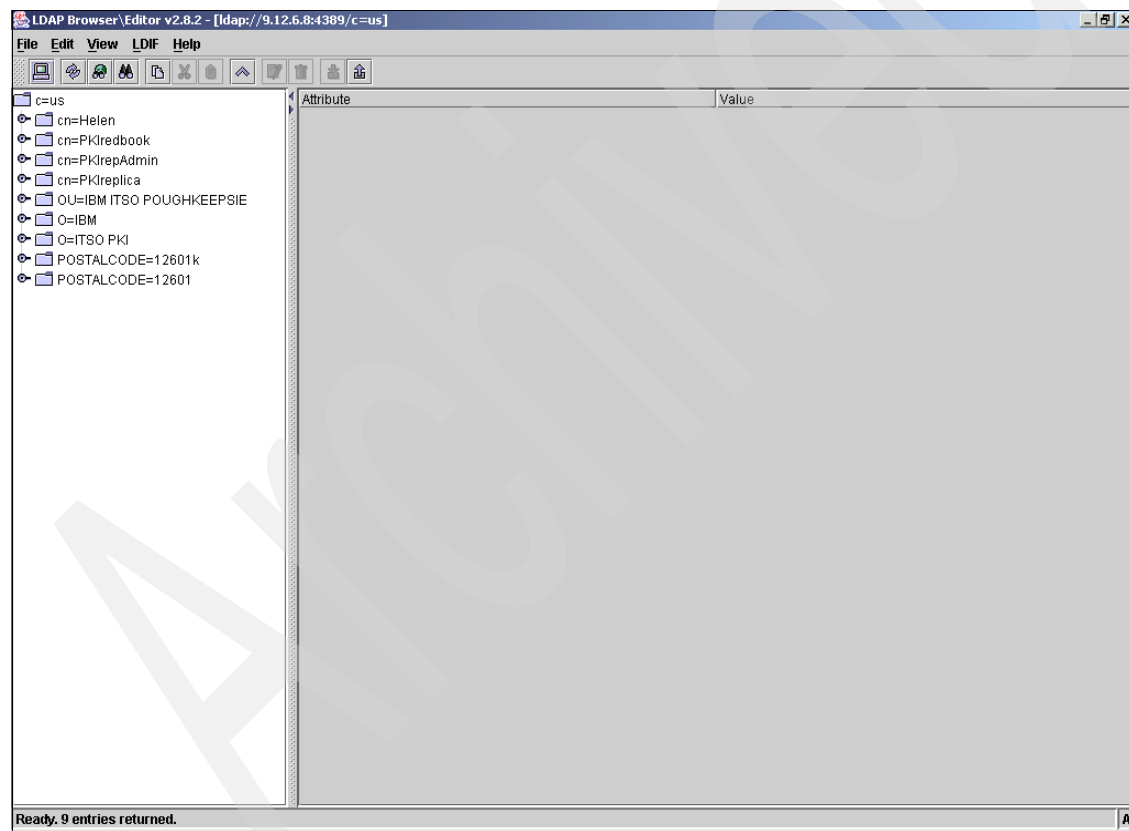


Figure 8-3 LDAPM before adding an entry

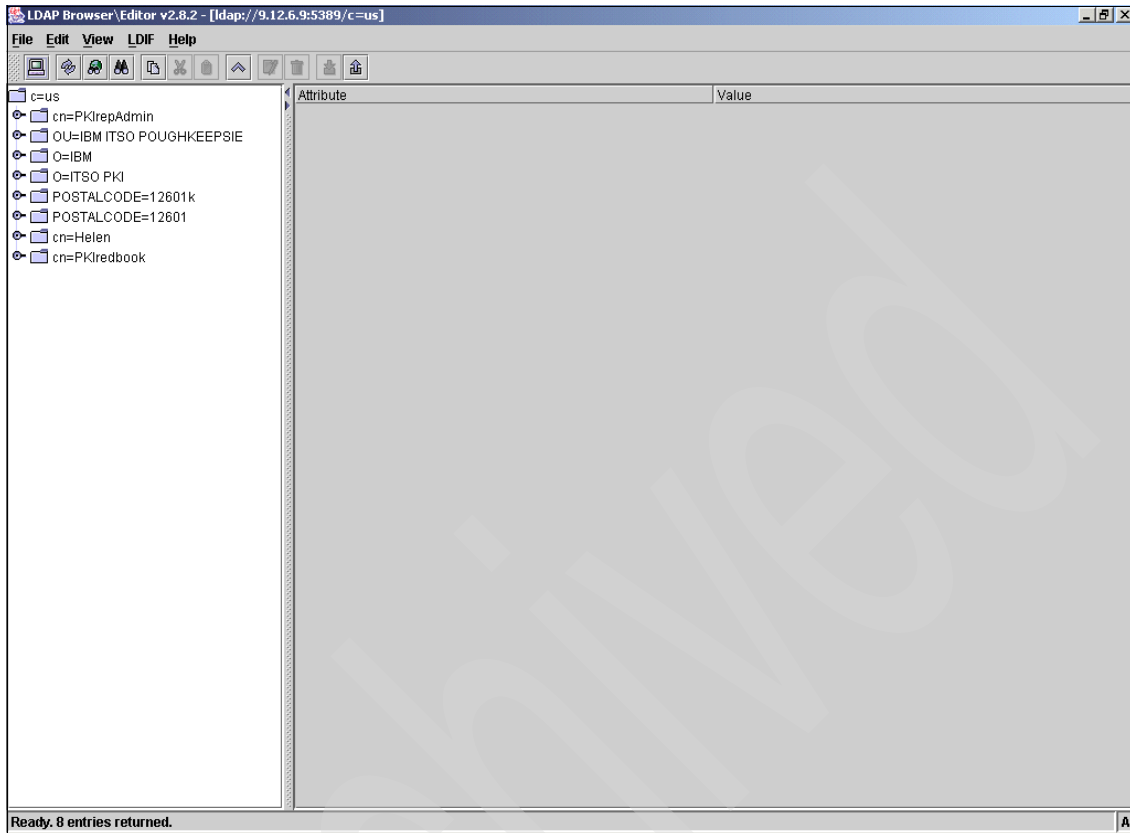


Figure 8-4 LDAPS before adding an entry

Then we ran the LDIF file shown in Example 8-6 from the UNIX command line in OMVS. This is to add a new entry called ITSO PKI book to our LDAP server.

Example 8-6 LDIF file to add entry ITSO PKI book

```
dn: cn=ITSO PKI book,c=us
objectclass: top
objectclass: person
cn: ITSO PKI book
sn: Chris rayns
```

Example 8-7 on page 247 shows the output from running this LDIF file.

Example 8-7 Output in OMVS from running chris.ldif

```
CHRISR:/u/chrisr: >cd /u/jjones/ldapstuff  
CHRISR:/u/jjones/ldapstuff: >ldapadd -h wtsc63.itso.ibm.com -p 4389 -D cn=Admin  
-w secret -f chris.ldif  
adding new entry cn=ITSO PKI book,c=us
```

```
CHRISR:/u/jjones/ldapstuff: >
```

We then looked at our LDAP servers again to see whether the entry had been added to both using replication. Figure 8-5 and Figure 8-6 on page 248 show that both LDAPM and LDAPS have the new entry, ITSO PKI book, so our replication works.

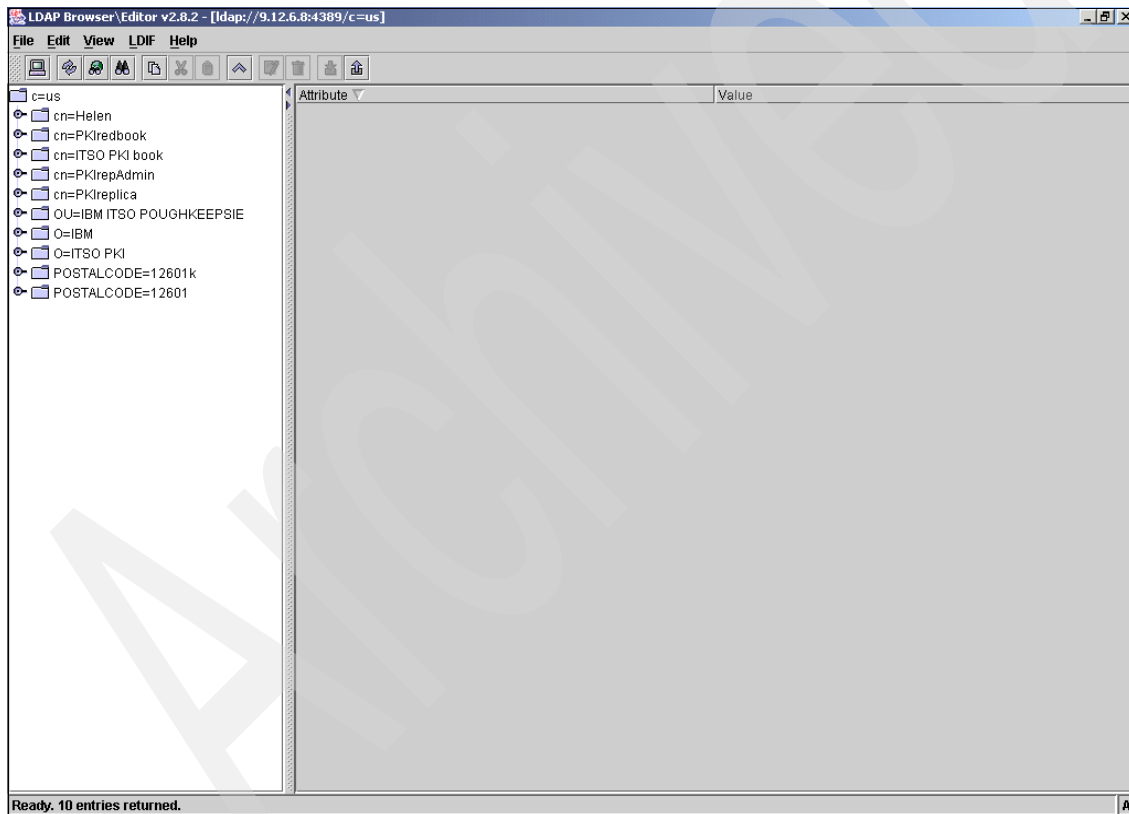


Figure 8-5 LDAPM(Primary) after addition of entry

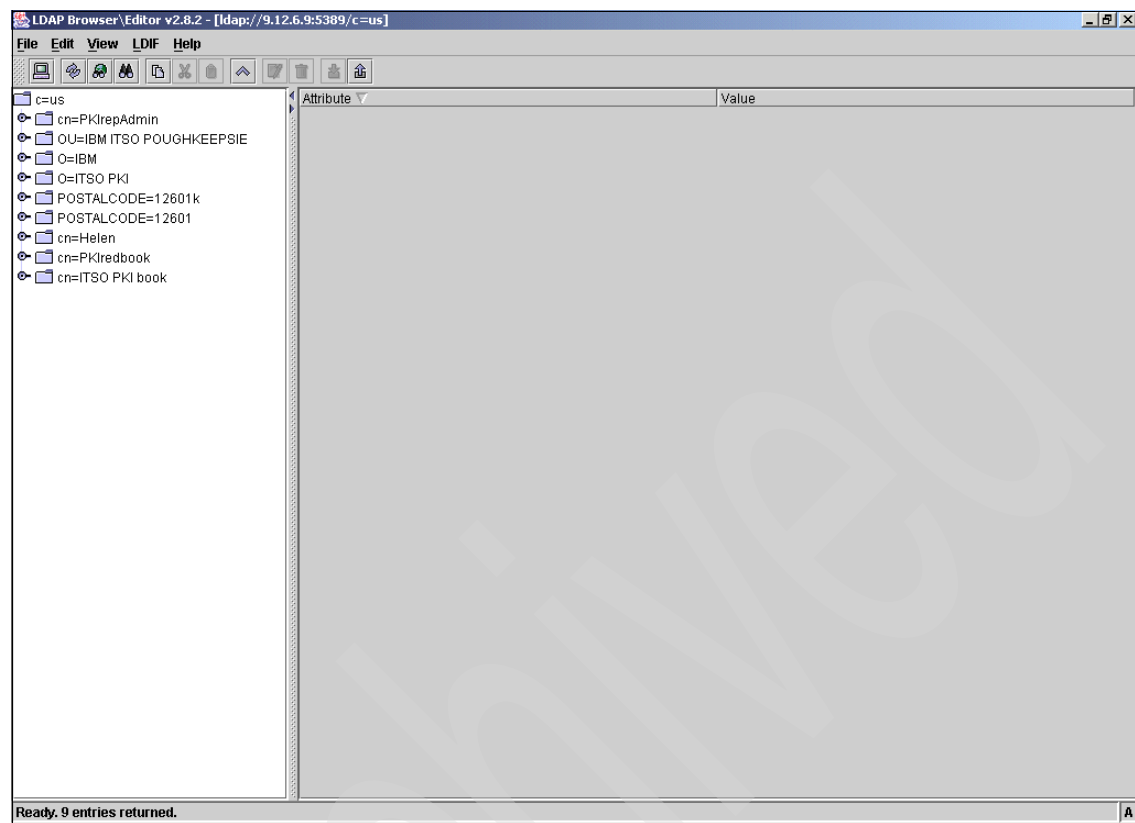


Figure 8-6 LDAPS(Replica) after addition

PKI Exit sample

Example: A-1 Sample PKI Exit

```

***** Top of Data *****
/*****
/*
/* COMPONENT_NAME: pkixit.c
/*
/* Licensed Materials - Property of IBM
/* 5694-A01
/* (C) Copyright IBM Corp. 2001
/* Status = HKY7706
/*
/*****
/*****
/*
/* This file contains sample code. IBM PROVIDES THIS CODE ON AN
/* 'AS IS' BASIS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR
/* IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
/* OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
/*
/*
/*****

#include <stdio.h>
#include <string.h>
#include <errno.h>

```

```

#include <stdlib.h>
#include <unistd.h>
#include <ctype.h>
#include <time.h>
#define GENCERT      "1"
#define REQCERT      "9"
#define EXPORT       "2"
#define REVOKE       "11"
#define GENRENEW      "12"
#define REQRENEW      "13"
#define PREEXIT       "0"

#define BUFSIZE 4096
#define CLASSMSK '\xC0'
#define CONSTMSK '\x20'
#define TAGMSK   '\x1F'
#define UTCTime  23
#define LONGLEN   '\x80'
#define LENBITS   '\x7F'

int preProcessGenReqCertExit(int argc, char* argv);
int postProcessGenReqCertExit(int argc, char* argv);
int preProcessExportExit(int argc, char* argv);
int postProcessExportExit(int argc, char* argv);
int preProcessRevokeExit(int argc, char* argv);
int postProcessRevokeExit(int argc, char* argv);
int preProcessGenReqRenewExit(int argc, char* argv);
int postProcessGenReqRenewExit(int argc, char* argv);

char *DERdecode(char *bstart, char *bend, int rclvl, char *notBefore, char
*notA
void copydate(char *ds, char *is, int len);
double determineExpiration(char *notAfter);

main(int argc, char * argv)
{
    int i, rc;

    if((strcmp(argv[2], GENCERT) == 0) | (strcmp(argv[2], REQCERT) == 0))
    {
        if (strcmp(argv[1], PREEXIT) == 0 )
        {
            rc = preProcessGenReqCertExit(argc, argv);
        }
        else
        {
            rc = postProcessGenReqCertExit(argc, argv);
        }
    }
}

```



```

else if((strcmp(argv2",EXPORT) == 0))
{
    if (strcmp(argv1",PREEXIT) == 0)
        rc = preProcessExportExit(argc, argv);
    else
        rc = postProcessExportExit(argc, argv);
}

else if((strcmp(argv2",REVOKE) == 0))
{
    if (strcmp(argv1",PREEXIT) == 0)
        rc = preProcessRevokeExit(argc, argv);
    else
        rc = postProcessRevokeExit(argc, argv);
}

else if ((strcmp(argv2",GENRENEW) == 0) | (strcmp(argv2",REQRENEW) == 0))
{
    if (strcmp(argv1",PREEXIT) == 0)
        rc = preProcessGenReqRenewExit(argc, argv);
    else
        rc = postProcessGenReqRenewExit(argc, argv);
}
else
    rc = -1;

return rc;
}

/*****
/*This function illustrates a GEN/REQCERT pre-exit can be used to
/*determine whether a GENCERT request can be granted by using RACF
/*authentication; and determine whether the request needs to be
/*approved by administrator.
/*Note: In order to use the call __check_resource_auth_np,
/*    this program needs to be program control.
/*
/*
/*Function: preProcessGenReqCertExit
/*    -if request is for authentication to z/OS, check resource
/*    access authorization as follows:
/*    -if client is a team member, allow the request and update
/*    the title for member and leader accordingly.
/*    -if client is a general partner, allow the request with
/*    admin approval, update the title.
/*    -if client is a trusted partner, allow the request with
/*    auto approval, update the title.
/*    -if neither a member nor a partner, deny the request.
/* Updated parameter(s): (You may update any parms to suit your need) */

```

```

/*          - Title=<new title>                                     */
/*****
int preProcessGenReqCertExit(int argc, char* argv[])
{
    int i, rc, needs_authentication = 0;
    char user[9]={'\0'};

    for (i=1; i<argc; i++)
    {
        if (strncmp(argv[i], "Template=", 9) == 0)
        {
            if (strstr(argv[i]+9,
                "PKI Browser Certificate For Authenticating To z/OS") != 0)
            {
                needs_authentication = 1;
            }
        }

        if (strncmp(argv[i], "UserId=", 7) == 0)
        {
            strcpy(user, argv[i]+7);
        }
    }

    /* if authentication is not necessary, do nothing */
    if (!needs_authentication)
    {
        return 0;
    }
    /* See if client is a member of the project*/
    rc = __check_resource_auth_np(NULL, NULL,
        user, "FACILITY", "PROJ.MEMBER", __READ_RESOURCE);

    if (rc == 0) /* client is a member */
    {
        /* Is client a leader of the project */
        if (0 == __check_resource_auth_np(NULL, NULL,
            user, "FACILITY", "PROJ.MEMBER", __UPDATE_RESOURCE))
        /* Update the Title parameter */
        printf("Title=Team Leader\n");
        else
        printf("Title=Team Member\n");

        return 0;
    }

    else /* See if client is a partner */
    {
        rc = __check_resource_auth_np(NULL, NULL,

```

```

        user,"FACILITY","PROJ.PARTNER",__READ_RESOURCE);
    }

    if (rc == 0 ) /* client is a partner */
    {
        /* See if he is a trusted partner */
        if (0 == __check_resource_auth_np(NULL,NULL,
            user,"FACILITY","PROJ.PARTNER",__UPDATE_RESOURCE))
        {
            /* Update the Title parameter */
            printf("Title=Team Trusted Partner\n");
            /* If it is a reqcert, change from admin approve to auto
            approve */
            if (strcmp(argv[2],REQCERT) == 0)
            {
                return 4;
            }
            else
            {
                return 0;
            }
        }
        else /* he is a general partner */
        {
            /* Update the Title parameter */
            printf("Title=Team Partner\n");

            /* If it is a gencert, change from auto approve to admin
            approve */
            if (strcmp(argv[2],GENCERT) == 0)
            {
                return 4;
            }
            else
            {
                return 0;
            }
        }
    } /* client is a partner */

    else /* Not a member nor partner, reject request */
    {
        return 8;
    }

}

/*****
/*This function illustrates a GEN/REQCERT post-exit can be used to */

```

```

/*transform the transaction id returned by PKI and turn it to a new */
/*form so that it can be used to retrieve the certificate stored */
/*in the customized repository, eg. a file or a database, later. */
/* */
/*Function: postProcessGenReqCertExit */
/*      -if request is for a browser certificate, prefix the */
/*      PKI and SAF transactionIds with 'PKI' and 'SAF' */
/*      accordingly. */
/* Updated parameter(s):(You may update any parms to suit your need) */
/*      - TransactionId */
/* */
/*****
int postProcessGenReqCertExit(int argc, char* argv)
{
    int i, needs_transform=0;
    char newPKITransid[60]="PKI";
    char newSAFTransid[60]="SAF";
    char originalTransid[60];

    strcpy(originalTransid,argv[argc-1]);

    for (i=1; i<argc; i++)
    {
        if (strcmp(argv[i],"Template=",9) == 0)
        {
            if (strstr(argv[i]+9, "Browser") != 0)
            {
                needs_transform = 1;
            }
        }
    }

    if (needs_transform)
    {
        /* If it is a PKI request (PKI transaction id starts with '1') */
        if (strcmp(originalTransid,"1",1) == 0)
        {
            strcat(newPKITransid, originalTransid);
            /* Update the TransactionId parameter */
            printf("%s\n",newPKITransid);
        }
        else
        {
            strcat(newSAFTransid, originalTransid);
            /* Update the TransactionId parameter */
            printf("%s\n",newSAFTransid);
        }
    }
}

```

```

    return 0;
}

/*****
/*This function illustrates an EXPORT pre-exit can be used to
/*recover the original transaction ids that have been transformed
/*by the GENCERT post exit; and reject the requests that haven't
/*gone through the transaction id transformation.
/*
/*
/*Function: preProcessExportExit
/*
/*    -if request is for a browser certificate and the
/*    transactionIds start with 'PKI' or 'SAF', take them out.
/*    If transactionId start with neither of them, reject the
/*    request.
/* Updated parameter(s):
/*    - TransactionId=<new transaction id>
/*
*****/
int preProcessExportExit(int argc, char* argv[])
{
    int i, needs_revert=0 ;
    char inputTransid[60];
    char originalTransid[60];

    for (i=1; i<argc; i++)
    {
        if (strcmp(argv[i], "Template=", 9) == 0)
        {
            if (strstr(argv[i]+9, "Browser") != 0)
            {
                needs_revert = 1;
            }
        }

        if (strcmp(argv[i], "TransactionId=", 14) == 0)
        {
            strcpy(inputTransid, argv[i]+14);
        }
    }

    if (needs_revert)
    {
        strcpy(originalTransid, &inputTransid[3]);

        /* If transactionid starts with "PKI" or "SAF" */
        if ((strcmp(inputTransid, "PKI", 3) == 0) |
            (strcmp(inputTransid, "SAF", 3) == 0))

```

```

    {
        /* Update the TransactionId parameter */
        printf("TransactionId=%s\n",originalTransid);
        return 0;
    }
    else /* reject the request */
    {
        return 8;
    }
}

else
{
    return 0;
}

}

/*****
/*This function illustrates an EXPORT post-exit can be used to
/*save the exported certificate in a customized repository, eg. a
/*file or a database.
/*
/*Function: postProcessExportExit
/*      -if request is for a server certificate, save the
/*      certificate in a file with transactionId as filename.
/*
*****/
int postProcessExportExit(int argc, char* argv[])
{
    int i, needs_saveToFile=0;
    FILE * fp;
    char filename[100]="/tmp/";

    for (i=1; i<argc; i++)
    {
        if (strcmp(argv[i],"Template=",9) == 0)
        {
            if ((strstr(argv[i]+9, "Server") != 0) |
                (strstr(argv[i]+9, "CA") != 0))
            {
                needs_saveToFile = 1;
            }
        }
    }
    if (strcmp(argv[i],"TransactionId=",14) == 0)
    {
        strcat(filename,argv[i]+14);
        strcat(filename,".crt");
    }
}

```

```

    }

    if (needs_saveToFile)
    {
        if((fp = fopen(filename, "w")) == NULL)
        {
            return (errno);
        }
        fprintf(fp, "%s\n", argv[argc-1]);
        fclose(fp);
    }

    return 0;
}

/*****
/*This function illustrates a REVOKE pre-exit can be used to
/*enforce the user to enter a revoke reason.
/*
/*
/*Function: preProcessRevokeExit
/*      -if revoke with no reason, reject the request.
/*
/*
*****/
int preProcessRevokeExit(int argc, char* argv[])
{
    int i ;

    for (i=1; i<argc; i++)
    {
        if (strncmp(argv[i], "reason=", 7) == 0)
        {
            /* If no revoke reason is given, reject the request */
            if (strcmp(argv[i]+7, "0") == 0)
            {
                return 8;
            }
            else
            {
                return 0;
            }
        }
    }
}

/*****
/*This function illustrates a REVOKE post-exit can be used to
/*save the revoke reasons in a customized repository, eg, a file or
/*a database for statistical use.
/*
*****/

```

```

/*Function: postProcessRevokeExit                                     */
/*      -save the revoke reason in a file.                           */
/*      */                                                           */
/*****/
int postProcessRevokeExit(int argc, char* argv)
{
    int i ;
    char filename = "/tmp/reasons.txt";
    FILE * fp;

    for (i=1; i<argc; i++)
    {
        if (strcmp(argv[i], "reason=") == 0)
        {
            if((fp = fopen(filename, "a")) == NULL)
                return (errno);

            fprintf(fp, "%s\n", argv[i]+7);
            fclose(fp);
        }
    }

    return 0;
}

/*****/
/*This function illustrates a GEN/REQ RENEW pre-exit can be used to */
/*determine how soon a certificate can be renewed; and adjust the */
/*end date of the certificate according to the renewal time.      */
/*      */                                                           */
/*Function: preProcessGenReqRenewExit                               */
/*      - extract the end date of the certificate                 */
/*      - determine how soon it will expire                      */
/*      - if it is more than 30 days before the expiration date, */
/*      reject the request.                                       */
/* Updated parameter(s): (You may update any parms to suit your need) */
/*      - NotAfter=<new notAfter period>                         */
/*      */                                                           */
/*****/
int preProcessGenReqRenewExit(int argc, char* argv)
{
    char * envString;
    char buffer[BUFSIZ], *bp;          /* Buffer to hold pem record */
    int cl, i;                        /* Misc */
    char notBefore[16];
    char notAfter[16];
    double timeBeforeExp; /* No. of secs before expire */
    char period[5] = {'\0'};
    int requestPeriod;

```



```

if((envString = getenv("HTTPS_CLIENT_CERT")) == NULL)
    return 51;

for (i=1; i<argc; i++)
{
    if (strcmp(argv[i], "NotAfter=", 9) == 0)
    {
        strcpy(period, argv[i]+9);
        requestPeriod = atoi(period);
    }
}

/* Decode the Certificate */
strcpy(buffer, envString);
cl= base64decode(buffer);
notAfter[0] = '\0';
notBefore[0] = '\0';
DERdecode(buffer, buffer+cl, 0, notBefore, notAfter);
timeBeforeExp = determineExpiration(notAfter);

/* Reject the request if it is more than 30 days before cert expires */
if (timeBeforeExp > 86400 * 30 )
    return 8;
else
{
    /* Update the NotAfter parameter */
    printf("NotAfter=%.0f\n", timeBeforeExp/86400 + requestPeriod);
    return 0;
}
}

/*****
/*This function illustrates a GEN/REQ RENEW post-exit can be used to
/*transform the transaction id returned by PKI and turn it to a new
/*form so that it can be used to retrieve the certificate stored
/*in the customized repository, eg. a file or a database, later.
/*
/*
/*Function: postProcessGenReqRenewExit
/*      -if request is for a browser certificate, prefix the
/*      PKI and SAF transactionIds with 'PKI' and 'SAF'
/*      accordingly.
/* Updated parameter(s):(You may update any parms to suit your need)
/*      - TransactionId
/*
*****/
int postProcessGenReqRenewExit(int argc, char* argv[])
{

```

```

int i, needs_transform=0;
char newPKITransid[60]="PKI";
char newSAFTransid[60]="SAF";
char originalTransid[60];

strcpy(originalTransid,argv[argc-1]);

for (i=1; i<argc; i++)
{
    if (strncmp(argv[i],"Template=",9) == 0)
    {
        if (strstr(argv[i]+9, "Browser") != 0)
        {
            needs_transform = 1;
        }
    }
}

if (needs_transform)
{
    /* If it is a PKI request */
    if (strncmp(originalTransid,"1",1) == 0)
    {
        strcat(newPKITransid, originalTransid);
        /* Update the Transaction Id parameter */
        printf("%s\n",newPKITransid);
    }
    else
    {
        strcat(newSAFTransid, originalTransid);
        /* Update the Transaction Id parameter */
        printf("%s\n",newSAFTransid);
    }
}

return 0;
}

/*****
/*Function: base64decode
/*          - decode a b64 encoded certificate
/*
/*****/
int base64decode(char *buffer)
{
    char outchar, *bp, *op;
    char trnchar[4];
    int numgroups, cl, i, l;
    /* Translation chars (6 bit values)
    /* Misc

```

```

/* Base64 Translation array. Each character maps to it's array index
   (starting from zero). The index is the translation character */

char base64Ŷ" =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"

numgroups= strlen(buffer) / 4;
l=0;
op=buffer; /* decode in place */
for (i=0,bp=buffer;i<numgroups;i++)
{
    trancharŶ0"= (char) (strchr(base64,*bp++) - base64);
    trancharŶ1"= (char) (strchr(base64,*bp++) - base64);
    outchar= (trancharŶ0" << 2) | (trancharŶ1" >> 4);
    *op++=outchar; l++;
    if (*bp != '=')
    {
        trancharŶ2"= (char) (strchr(base64,*bp++) - base64);
        outchar= (trancharŶ1" << 4) | (trancharŶ2" >> 2);
        *op++=outchar; l++;
        if (*bp != '=')
        {
            trancharŶ3"= (char) (strchr(base64,*bp++) - base64);
            outchar= (trancharŶ2" << 6) | trancharŶ3";
            *op++=outchar; l++;
        }
    }
}
return(l);
}

/*****
/*Function: DERdecode                                     */
/*          - decode a DER encoded certificate             */
/*                                                     */
/*****
char *DERdecode(char *bstart, char *bend, int rclvl, char *notBefore, char
*notA
{
    /* Do the DER decoding from bstart to bend where rclvl is the
       current recursion level */

    char *p, c;
    int lenbytes, i;
    unsigned ui;

    int cls, /* BER encoding CLASS */
        pc, /* " " Primitive/Constructed Flag 1=Constructed */
        tag, /* " " Tag */

```

```

        len; /* " " Length */

p= bstart;
while((strlen(notAfter) == 0) && (p < bend-1) && ((*p != 0) || (*(p+1) !=
0)))
{
    /* Get the class from the high order 2 bit of the first byte */
    cls= *p & CLASSMSK;
    /* Now get the constructed flag from the third bit */
    pc= *p & CONSTMSK;
    /* Determine the tag from the low order 5 bits */
    tag= *p & TAGMSK;
    p++; /* Done with the class byte */
    /* Determine the BER length. The high order bit being "OFF" indicates
       that the length byte represents the complete length (short form max
       127). The high order bit being "ON" represents long form length, in
       which case the length byte represents the number of subsequent octets
       that represent the length (i.e., the length of the length). A length
       byte of x'80' (long form, zero length octets) indicates indefinite
       length. In in case the data octets are terminated with a double NULL
       byte. */

    if ((*p & LONGLEN) == 0)
    {
        /* Short form length */
        len= *p & LENBITS;
        p++; /* done with the length byte */
    }
    else
    {
        /* long form length */
        lenbytes= *p & LENBITS;
        p++;
        len= 0;
        for (i=0 ; i < lenbytes ; i++,p++)
        {
            len *= 256;
            ui= *p;
            ui= ui & 0x00FF;
            len += ui;
        }
    }
    /* Get the actual DER data to locate the dates */
    if (pc) /* Constructed data */
    {
        p= DERdecode(p,p+len,rclvl+1,notBefore,notAfter);
    }
    else /* Primitive data */
    {

```

```

switch (tag)
{
    case UTCTime:
        if (strlen(notBefore) == 0)
            copydate(notBefore,p,len);
        else
            copydate(notAfter,p,len);
        break;
    default:
        /* We don't care about any other data type */
        break;
}
p += len;
}
}
return(p); /* p points to the next DER element */
}

/*****
/*Function: DERdecode
/*      - copy the UTC datetime and convert to EBCDIC
/*
/*****/
void copydate(char *ds,char *is,int len)
{
    int i;
    if (*is >= 0x35)
        strcpy(ds,"19");
    else
        strcpy(ds,"20");
    ds += 2;
    for (i=0;i<len;i++,ds++,is++)
        if (*is == 0x5a)
            *ds= 'Z';
        else
            *ds= '0' + (*is - 0x30);
    *ds= '\0';
}

/*****/
/*Function: determineExpiration
/*      - find out the time before the cert expired
/*
/*****/
double determineExpiration(char* notAfter)
{
    time_t tt; /* notAfter converted to a time_t */
    struct tm ts, *t2;
    struct tm *tmpTm;

```

```

time_t      now, gmNow;
double      timeBeforeExp;

sscanf(notAfter,
        "%4d%2d%2d%2d%2dZ",
        &ts.tm_year,
        &ts.tm_mon,
        &ts.tm_mday,
        &ts.tm_hour,
        &ts.tm_min,
        &ts.tm_sec);

/*determine Zulu time (now), and Zulu time + our timezone (gmNow) */
now = time(NULL);
tmpTm= gmtime(&now);
tmpTm->tm_isdst = 0;    /* don't figure in daylight savings */
gmNow = mktime(tmpTm);

/* mktime() assumes local time, so adjust our UTC to local time */
ts.tm_min += ((int)difftime(now, gmNow) / 60);
ts.tm_year -= 1900;    /* tm_years is years since 1900 */
ts.tm_mon -= 1;        /* tm_mon is zero based */

ts.tm_isdst = 0;    /* don't figure in daylight savings */

tt = mktime(&ts);
/* tt now contains the notAfter date/time */
t2= localtime(&tt);

timeBeforeExp = difftime(tt, now);

return (timeBeforeExp);
}

```



List of sample files provided with PKI Services

This appendix illustrates some examples provided with PKI Services.

httpd.conf sample for PKI Web server 1

This example shows the contents of /usr/lpp/pkiserv/samples/httpd.conf.

Example: B-1 /usr/lpp/pkiserv/samples/httpd.conf

```
# Licensed Materials - Property of IBM
# 5694-A01
# (C) Copyright IBM Corp. 2001
# Status = HKY7706

# For a secure system, set the default User ID to %%CLIENT%%
UserId      %%CLIENT%%

# SSL support using a SAF keyring
keyfile SSLring SAF
# OR
# May use a gskkyman key database instead of SAF keyring
#keyfile /etc/key.kdb

sslmode on
sslport 443
Normalmode on
Protection PublicUser {
    ServerId      PublicUser
    UserID        PKISERV
    Mask          Anyone
}
Protect /PKIServ/public-cgi/* PublicUser
Protect /PKIServ/ssl-cgi-bin/* PublicUser
Protect /PKIServ/* PublicUser

Protection AuthenticatedUser {
    ServerId      AuthenticatedUser
    AuthType      Basic
    PasswdFile    %%SAF%%
    UserID        %%CLIENT%%
    Mask          All
}
Protect /PKIServ/ssl-cgi-bin/auth/* AuthenticatedUser

Protection SurrogateUser {
    ServerId      SurrogateUser
    AuthType      Basic
    PasswdFile    %%SAF%%
    UserID        PKISERV
    Mask          All
}
```



```

Protect /PKIServ/ssl-cgi-bin/surrogateauth/* SurrogateUser

Redirect /PKIServ/ssl-cgi/* https://<server-domain-name>/PKIServ/ssl-cgi-bin/*
Redirect /PKIServ/ssl-cgi/auth/* https://<server-domain-name>/PKIServ/ssl-cgi-bin/auth/*
Redirect /PKIServ/ssl-cgi/surrogateauth/*
https://<server-domain-name>/PKIServ/ssl-cgi-bin/surrogateauth/*

Redirect /PKIServ/clientauth-cgi/* https://<server-domain-name>:1443/PKIServ/clientauth-cgi/*

Exec /PKIServ/public-cgi/* <application-root>/PKIServ/public-cgi/*
Exec /PKIServ/ssl-cgi-bin/* <application-root>/PKIServ/ssl-cgi-bin/*
Pass /PKIServ/cacerts/* /var/pkiserv/*

AddType .cer application/x-x509-user-cert ebcdic 0.5 # Browser Certificate
AddType .der application/x-x509-ca-cert binary 1.0 # CA Certificate

```

httpd.envvars sample for the PKI Web server

This example shows the contents of `/usr/lpp/pkiserv/samples/httpd.envvars`.

Example: B-2 /usr/lpp/pkiserv/samples/httpd.envvars

```

# Licensed Materials - Property of IBM
# 5694-A01
# (C) Copyright IBM Corp. 2001
# Status = HKY7706

_PKISERV_CONFIG_PATH=/etc/pkiserv
#_PKISERV_EXIT=/<full-path-to-pkiexit>/pkiexit

```

httpd.conf sample for PKI Web server 2

This example shows the contents of `/usr/lpp/pkiserv/samples/httpd2.conf`.

Example: B-3 /usr/lpp/pkiserv/samples/httpd2.conf

```

# Licensed Materials - Property of IBM
# 5694-A01
# (C) Copyright IBM Corp. 2001
# Status = HKY7706

# For a secure system, set the default User ID to %%CLIENT%%
UserId      %%CLIENT%%

```

```

# SSL support using a SAF keyring
keyfile SSLring SAF
#
# OR
# May use a gskkyman key database instead of SAF keyring
#keyfile /etc/key.kdb

sslmode on
sslport 1443
Normalmode off
SSLClientAuth strong
SSLX500CARoots local_and_x500
SSLX500Host <ldap-server-name>
SSLX500Port <ldap-port-number>
SSLX500UserID <ldap-distinguished-name>
SSLX500Password <ldap-password>

Protection RenewRevokeUser {
    ServerId      RenewRevokeUser
    AuthType      Basic
    UserID        PKISERV
    SSL_CLIENTAUTH Client
    Mask          Anyone
}

Protect /PKIServ/clientauth-cgi/* RenewRevokeUser

Protection AuthenticatedAdmin {
    ServerId      AuthenticatedAdmin
    AuthType      Basic

    UserID        %%CERTIF%%
    SSL_CLIENTAUTH Client
    Mask          Anyone
}

Protect /PKIServ/clientauth-cgi/auth/* AuthenticatedAdmin

Redirect /PKIServ/public-cgi/* http://<server-domain-name>/PKIServ/public-cgi/*
Redirect /PKIServ/ssl-cgi/* https://<server-domain-name>/PKIServ/ssl-cgi-bin/*

Exec /PKIServ/clientauth-cgi/* <application-root>/PKIServ/clientauth-cgi-bin/*

```

pkiserv.conf

Example: B-4 pkiserv.conf

Licensed Materials - Property of IBM

```
# 5694-A01
# (C) Copyright IBM Corp. 2001, 2002
# Status = HKY7707
```

```
[OIDs]
C=2.5.4.6
O=2.5.4.10
OU=2.5.4.11
CN=2.5.4.3
L=2.5.4.7
ST=2.5.4.8
TITLE=2.5.4.12
POSTALCODE=2.5.4.17
STREET=2.5.4.9
MAIL=0.9.2342.19200300.100.1.3
sha-1WithRSAEncryption=1.2.840.113549.1.1.5
id-dsa-with-sha1=1.2.840.10040.4.3
MyPolicy=1.2.3.4
```

```
[ObjectStore]
# Data set name of the VSAM request (object store) base CLUSTER
ObjectDSN='pkisrvd.vsam.ost'

# Data set name of the VSAM object store PATH for the transaction ID (TID) alternate index
ObjectTidDSN='pkisrvd.vsam.ost.path'

# Data set name of the VSAM object store PATH for the status alternate index
ObjectStatusDSN='pkisrvd.vsam.ost.status'

# Data set name of the VSAM object store PATH for the requestor alternate index
ObjectRequestorDSN='pkisrvd.vsam.ost.requestr'

# Data set name of the VSAM issued certificate list (ICL) base CLUSTER
ICLDSN='pkisrvd.vsam.icl'

# Data set name of the VSAM ICL PATH for the status alternate index
ICLStatusDSN='pkisrvd.vsam.icl.status'

# Data set name of the VSAM ICL PATH for the requestor alternate index
ICLRequestorDSN='pkisrvd.vsam.icl.requestr'

# How days (d) or weeks (w) should completed requests remain in the object store?
# Specify 0d to indicate completed requests should not be removed
RemoveCompletedReqs=1w

# How days (d) or weeks (w) should inactive requests remain in the object store?
# Specify 0d to indicate inactive requests should not be removed
RemoveInactiveReqs=4w
```

```

# How many days (d) or weeks (w) should expired certificates remain in the ICL?
# Specify 0d to indicate expired certificates should not be removed
#RemoveExpiredCerts=26w

# Are the VSAM data sets shared in a sysplex with other instances
# of PKI Services. True (T) or False (F)
SharedVSAM=F

[CertPolicy]
SigAlg1=sha-1WithRSAEncryption
CreateInterval=3m

# When the warning message should be issued. (i.e. the number of days
# or weeks before the certificate expiration date/time). Defaults to never
ExpireWarningTime=4w

TimeBetweenCRLs=1d
CRLDuration=2d

# Maximum number of certificates that may appear on one distribution point CRL.
# The default is 0 which indicates distribution point CRLs should not be created.
CRLDistSize=500

# Constant portion of the CRL distribution point leaf-node relative distinguished name.
# The distribution point number is appended to this value to form the common name.
# The default value is "CRL".
CRLDistName=CRL

PolicyRequired=F
PolicyCritical=F

PolicyName1=MyPolicy
Policy1Org=MyOrganization
Policy1Notice1=3
Policy1Notice2=17
UserNoticeText1=This is some very lawyerly statement for the relying party to read and make
decisions based on.
CPS1=http://www.mycompany.com/cps.html

[General]
InitialThreadCount=10

# full pathname or data set name containing the 'your certificate is ready'
# message form. Defaults to no message issued
ReadyMessageForm=/etc/pkiserv/readymsg.form

# full pathname or data set name containing the 'your certificate request
# has been rejected' message form. Defaults to no message issued

```

```
RejectMessageForm=/etc/pkiserv/rejectmsg.form
```

```
# full pathname or data set name containing the 'your certificate is about  
# to expire' message form. Defaults to no message issued  
ExpiringMessageForm=/etc/pkiserv/expiringmsg.form
```

```
[SAF]
```

```
KeyRing=PKISRVD/CAring
```

```
[LDAP]
```

```
NumServers=1
```

```
PostInterval=5m
```

```
Server1=myldapserver.mycompany.com:389
```

```
AuthName1=CN=root
```

```
AuthPwd1=root
```

```
CreateOUValue= Created by PKI Services
```

```
RetryMissingSuffix=T
```

```
# Name of the LDAPBIND Class profile containing the bind information for LDAP  
# server 1. This key is optional. Used in place of keys Server1, AuthName1.  
# and AuthPwd1
```

```
#BindProfile1=LOCALPKI.BINDINFO.LDAP1
```

pkiserv.envars

Example: B-5 pkiserv.envars

```
#-----#  
#  
# PKI Services sample environment variable file  
#  
# Licensed Materials - Property of IBM  
# 5694-A01  
# (C) Copyright IBM Corp. 2001, 2002  
# Status = HKY7707  
#  
#-----#  
#  
# Language and Path configurations  
#  
LANG=En_US.IBM-1047  
PATH=/usr/sbin  
LIBPATH=/usr/lpp/pkiserv/lib:/usr/lib  
NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lpp/pkiserv/lib/nls/msg/%L/%N  
#  
# Configuration File location and Message configuration Options  
#
```

```

_PKISERV_CONFIG_PATH=/etc/pkiserv
_PKISERV_MSG_LOGGING=stdout_logging
_PKISERV_MSG_LEVEL=*.w
#
# Location of the OCSF Registry (/var/ocsf is the default location)
#
OCSFREGDIR=/var/ocsf

```

pkiserv.tmpl

Example: B-6 pkiserv.tmpl

```

#
# =====
#
# COMPONENT_NAME: pkiserv.tmpl
#
# Licensed Materials - Property of IBM
# 5694-A01
# (C) Copyright IBM Corp. 2001, 2002
# Status = HKY7707
#
# =====
# Change-Activity:
# $D1=OW55613, HKY7707, 020626
# $D2=OW57109, HKY7707, 021114
# $D3=OW57573, HKY7706, 021217
#
# Change Descriptions:
# C - Use AltEmail in S/MIME template @D1A
# C - No change - reship @D2A
# C - Changed xenroll ActiveX control for MS patch @D3A
#
# =====
#
# Configuration file for interfacing with R_PKIServ. This file may be
# customized as required by the installation. Any line with an '#' in
# column 1 is considered a comment.
#
# Structure:
#
# The file contains a mixture of true HTML and HTML like tags. The
# main tags divide the file into sections, APPLICATION, TEMPLATE,
# and INSERT, where APPLICATION and TEMPLATE may contain various
# subsections, named fields, and substitution variables as explained
# below.
#
#

```

```

# <APPLICATION NAME=appl-name> ... </APPLICATION>
#
# This section identifies the applications that will make use of
# PKI Services for Z/OS. The product ships with one application
# defined, "PKISERV". This section may contain the following subsections:
#
# <CONTENT> ... </CONTENT>
#
# This subsection contains the HTML to be presented to the end
# user requesting and retrieving certificates
#
# The subsection should contain one or more named fields
# identifying certificate templates to be used for requesting
# or managing certificates through this application. (See below
# for a description of named fields.) These template names
# should match the HTML selection value associated with them.
#
# <RECONTENT> ... </RECONTENT>
#
# This subsection contains the HTML which will display the
# certificate details so that the end user may confirm that
# that is the certificate to be renewed or revoked. This will
# make use of a new substitution variable, [printablecert],
# which contains the data extracted from the ICL entry.
#
# <RESUCCESSCONTENT> ... </RESUCCESSCONTENT>
#
# This subsection contains the HTML to be presented to the end
# user when the certificate revoke request
# was submitted successfully.
# Any named fields in this subsection are interpreted as
# content inserts defined by INSERT sections. For PKISERV, the
# INSERT sections are included as part of the HTML presented
# to the end user.
#
# <REFAILURECONTENT> ... </REFAILURECONTENT>
#
# This subsection contains the HTML to be presented to the end
# user when the certificate renew/revoke request submit failed.
# Any named fields in this subsection are interpreted as
# content inserts defined by INSERT sections. For PKISERV, the
# INSERT sections are included as part of the HTML presented
# to the end user.
#
# <ADMINHEADER> ... </ADMINHEADER>
#
# This subsection contains the general installation specific HTML
# content for the header of all admin pages.
#

```

```

# <ADMINFOOTER> ... </ADMINFOOTER>
#
# This subsection contains the general installation specific HTML
# content for the footer of all admin pages.
#
#
# <TEMPLATE NAME=tmp1-name> ... </TEMPLATE>
# <TEMPLATE NAME=tmp1-name alias>
# <NICKNAME=nick-name>
#
# This section defines the certificate templates referenced in the
# APPLICATION sections. You may refer to a single template by
# more than one name using aliases. Also, as the template name
# must be recalled in order to renew a certificate, it
# must be stored with the certificate. The nickname of the
# template will serve this purpose.
#
# Applicable subsections are:
#
# <CONTENT> ... </CONTENT>
#
# This subsection contains the HTML to be presented to the end
# user requesting certificates of this type. Any named fields
# in this subsection are interpreted as certificate field names
# defined by INSERT sections. (See below for a description of
# named fields.) For PKISERV, the INSERT sections
# are included as part of the HTML presented to the end user.
# (i.e., the end user provides values for these fields.)
# Named fields in this subsection are considered optional if
# the named field contains more than one word within the %%
# delimiters, e.g., %%AltName (Optional)%%. The user need not
# supply a value for AltName
#
# <APPL> ... </APPL>
#
# This subsection identifies certificate fields that the
# application itself should provide values for. This subsection
# should contain named fields only, one per line. Currently,
# the only supported named field allowed in this section is
# "UserId"
#
# <CONSTANT> ... </CONSTANT>
#
# This subsection identifies certificate fields that have a
# constant (hardcoded) value for everyone. This subsection
# should contain named fields only, one per line. The syntax
# for specifying the values is %%field-name=field-value%%,
# e.g., %%KeyUsage=handshake%%
#
#

```



```

# <SUCCESSCONTENT> ... </SUCCESSCONTENT>
#
# This subsection contains the HTML to be presented to the end
# user when the certificate request was submitted successfully.
# Any named fields in this subsection are interpreted as
# content inserts defined by INSERT sections. For PKISERV, the
# INSERT sections are included as part of the HTML presented
# to the end user.
#
# <FAILURECONTENT> ... </FAILURECONTENT>
#
# This subsection contains the HTML to be presented to the end
# user when the certificate request submit failed.
# Any named fields in this subsection are interpreted as
# content inserts defined by INSERT sections. For PKISERV, the
# INSERT sections are included as part of the HTML presented
# to the end user.
#
# <RETRIEVECONTENT> ... </RETRIEVECONTENT>
#
# This subsection contains the HTML to be presented to the end
# user to enable certificate retrieval.
# Any named fields in this subsection are interpreted as
# content inserts defined by INSERT sections. For PKISERV, the
# INSERT sections are included as part of the HTML presented
# to the end user.
#
# <RETURNCERT> ... </RETURNCERT>
#
# This subsection contains the HTML to be presented to the
# enduser upon successful certificate retrieval. For PKISERV, if
# the certificate being retrieved is a browser certificate, then
# this section must contain a single line containing a browser
# qualified INSERT name, e.g., %returnbrowsercert
# [browser type]%. Additionally, INSERTs for Netscape
# (returnbrowsercertNS) and Internet Explorer (returnbrowsercertIE)
# containing browser specific HTML for returning certificates must
# be defined elsewhere in the configuration file. If the
# certificate being retrieved is a server certificate, this section
# should contain the HTML necessary to present the certificate
# to the user as text
#
# <INSERT NAME=insert-name> ... </INSERT>
# This section contains HTML that either describes a certificate
# field or defines other common HTML that may be referenced in
# the TEMPLATE sections. INSERTs are referenced elsewhere by
# using a named field of the form %insert-name%
#
# Named Fields - Delineated with %, e.g., %Label%. Their meaning

```

```

# is specific to the section they are contained in. Named fields
# are case sensitive. Named fields are also used to reference common
# includeable HTML. Note, PKISERV treats named fields that begin with
# a dash as just includeable code. Any special meaning a named field
# may have, given the section it's contained in, is ignored if it
# begins with a dash. For example, if %%-pagefooter%% was specified
# in a TEMPLATE CONTENT section, -pagefooter would not be considered
# a certificate field name. However, the INSERT with the name
# -pagefooter would be included in the HTML page presented to the
# end user.
#
# Substitution Variables - Delineated with square brackets, e.g.,
# [base64cert]. They represent variables that get replaced with
# an actual value at run time. Substitution variables are case
# sensitive. The valid substitution variables are:
#
#   transactionid - Unique value returned from a certificate request.
#
#   tmplname - Certificate template name. Primed from the HTML tag
#   <SELECT NAME="Template"> in the <APPLICATION NAME=PKISERV>
#   section. This is selected by the end user on the first web page.
#
#   iecert - The requested certificate in a form the Microsoft
#   Internet Explorer accepts.
#
#   base64cert - The requested base64 encoded certificate.
#
#   browsertype - Special substitution variable to be used to qualify
#   named field only. Its use enables the different browsers,
#   Netscape and Internet Explorer, to perform browser specific
#   operations, i.e., Netscape uses a KEYGEN HTML tag to generate a
#   public/private key pair while Internet Explorer uses ACTIVEX
#   controls. For example, if %%PublicKey[browsertype]%% was
#   specified in a TEMPLATE CONTENT section referenced by a user
#   with the Netscape Navigator browser then INSERT PublicKeyNS
#   would be included. Likewise, if the user's browser was the
#   Microsoft Internet Explorer, INSERT PublicKeyIE would be included.
#
#   optfield - Special substitution variable that should be placed in
#   any certificate field name INSERT where the value may be supplied
#   by the end user. It enables the field to be displayed as optional
#   if desired.
#
#   printablecert - Summary information about the certificate to be
#   renewed/revoked, such as issuer's name, subject's
#   name...
#
#   errorinfo - Information about the failing SAF call such as the
#   return code and reason code.

```

```

#
# Note, depending on where a substitution variable is used, it may
# not have a valid meaning, e.g., base64cert would be meaningless
# prior to the certificate being retrieved. The value of
# [base64cert] would be the empty string (aka NULL) in this case.
#
# =====
#
# Application - PKISERV
#
# The installation should customize the CONTENT and ADMINCONTENT
# subsections as appropriate
#
# =====
#
<APPLICATION NAME=PKISERV>
<CONTENT>
<HTML><HEAD>
<TITLE> Web Based Certificate Generation Application </TITLE>
%%-copyright%%
</HEAD>
<BODY>
<H1>PKI Services Certificate Generation Application</H1>
<p>
<A HREF="/PKIServ/cacerts/cacert.der">Install
  our CA certificate into your browser </A>
<H2>Choose one of the following:</H2>
<ul>
<li><h3>Request a new certificate using a model</h3>
<FORM name=mainform METHOD=GET ACTION="/PKIServ/ssl-cgi/catmpl.rexx">
<p> Select the certificate template to use as a model
<SELECT NAME="Template">
  %%1 Year PKI SSL Browser Certificate%%
    <OPTION>1 Year PKI SSL Browser Certificate
  %%1 Year PKI S/MIME Browser Certificate%%
    <OPTION>1 Year PKI S/MIME Browser Certificate
  %%2 Year PKI Browser Certificate For Authenticating To z/OS%%
    <OPTION>2 Year PKI Browser Certificate For Authenticating To z/OS
  %%5 Year PKI SSL Server Certificate%%
    <OPTION>5 Year PKI SSL Server Certificate
  %%5 Year PKI IPSEC Server (Firewall) Certificate%%
    <OPTION>5 Year PKI IPSEC Server (Firewall) Certificate
  %%5 Year PKI Intermediate CA Certificate%%
    <OPTION>5 Year PKI Intermediate CA Certificate
  %%1 Year SAF Browser Certificate%%
    <OPTION>1 Year SAF Browser Certificate
  %%1 Year SAF Server Certificate%%
    <OPTION>1 Year SAF Server Certificate
</SELECT>

```

```

<p>
<INPUT TYPE="submit" VALUE="Request Certificate">
</FORM>
<li><h3>Pick up a previously requested certificate</h3>
<FORM name=selfform METHOD=GET ACTION="/PKIServ/ssl-cgi/caretrieve.rexx">
<p> Enter the assigned transaction ID
<INPUT NAME="TransactionId" TYPE="text" SIZE=56 maxlength="56">
<br>Select the certificate return type
<SELECT NAME="Template">
%%PKI Browser Certificate%%
    <OPTION>PKI Browser Certificate
%%PKI Server Certificate%%
    <OPTION>PKI Server Certificate
%%SAF Browser Certificate%%
    <OPTION>SAF Browser Certificate
%%SAF Server Certificate%%
    <OPTION>SAF Server Certificate
</SELECT>
<p>
<INPUT TYPE="submit" VALUE="Pick up Certificate">
</FORM>
<li><h3>Renew or revoke a previously issued browser certificate</h3>
<FORM name=selfform METHOD=GET ACTION="/PKIServ/clientauth-cgi/cadisplay.rexx">
<p>
<SCRIPT LANGUAGE="JavaScript">
<!--
function RenewRevokeAlert(){
var STRING_RenewRevokePrompt=
    "You will be prompted by the browser to select " +
    "the certificate you want to renew or revoke. " +
    "Once you select the certificate you will be " +
    "given the opportunity to confirm your selection. " +
    "Note that you can only renew or revoke a single " +
    "certificate per one browser session. If you wish " +
    "to renew or revoke another certificate, you must " +
    "close your browser and restart it.";
    alert(STRING_RenewRevokePrompt);
    return true;
}

function ValidateEntry(){
var STRING_MissingFieldPrompt=
    "Enter the required field."
var STRING_MissingConfirmPwdPrompt=
    "Reenter password."
var STRING_UnmatchPwdPrompt=
    "The passwords do not match. Enter again."
if(document.renform.PassPhrase.value=="") {
    alert(STRING_MissingFieldPrompt);

```

```

document.renform.PassPhrase.focus();
return true;
}
else if(document.renform.ConfirmPassPhrase.value=="") {
alert(String_MissingConfirmPwdPrompt);
document.renform.ConfirmPassPhrase.focus();
return true;
}
else if(document.renform.PassPhrase.value!=
document.renform.ConfirmPassPhrase.value){
alert(String_UnmatchPwdPrompt);
document.renform.ConfirmPassPhrase.focus();
return true;
}
else {
return false;
}
}
}
//-->
</SCRIPT>
<INPUT TYPE="submit" VALUE="Renew or Revoke Certificate"
onClick="return RenewRevokeAlert()">
</FORM>
<li><h3>Administrators click here</h3>
# The following action will force userid/pw authentication for administrators
<FORM name=admform METHOD=GET ACTION="/PKIServ/ssl-cgi/auth/admmain.rexx">
# The following action will force client certificate authentication for administrators
#<FORM name=admform METHOD=GET
# ACTION="/PKIServ/clientauth-cgi/auth/admmain.rexx">
<p>
<INPUT TYPE="submit" VALUE="Go to Administration Page">
</FORM>
</ul>
<p> %%-pagefooter%%
</BODY>
</HTML>
</CONTENT>
<RECONTENT>
<HTML><HEAD>
<TITLE> PKISERV Renew or Revoke a Browser Certificate </TITLE>
%%-copyright%%
</HEAD>
<BODY>
<H1>Renew or Revoke a Browser Certificate</H1>
<h3>Here is the certificate you selected:</h3>
<p>
[printablecert]
<h2>If this is the correct certificate, choose one of the following:</h2>
<b>(otherwise you need to restart your browser to pick another certificate)</b>

```

```

<ul>
<h3><li>Renew the above certificate</h3>
<FORM name=renform METHOD=POST
  ACTION="/PKIServ/clientauth-cgi/camodify.rexx">

<SCRIPT LANGUAGE="JavaScript">
<!--

function ValidateEntry(){
var STRING_MissingFieldPrompt=
    "Enter the required field."
var STRING_MissingConfirmPwdPrompt=
    "Reenter password."
var STRING_UnmatchPwdPrompt=
    "The passwords do not match. Enter again."
if(document.renform.PassPhrase.value=="") {
alert(STRING_MissingFieldPrompt);
document.renform.PassPhrase.focus();
return true;
}
else if(document.renform.ConfirmPassPhrase.value=="") {
alert(STRING_MissingConfirmPwdPrompt);
document.renform.ConfirmPassPhrase.focus();
return true;
}
else if(document.renform.PassPhrase.value!=
    document.renform.ConfirmPassPhrase.value){
alert(STRING_UnmatchPwdPrompt);
document.renform.ConfirmPassPhrase.focus();
return true;
}
else {
return false;
}
}
//-->
</SCRIPT>
<INPUT NAME="action" TYPE="hidden" VALUE="renew">
%%NotifyEmail (optional)%%
%%PassPhrase%%
<p>
<INPUT TYPE="submit" VALUE="Renew" onClick=
  "if(ValidateEntry()) return false; else return true;">
</FORM>
<h3><li>Revoke the above certificate</h3>
<FORM name=revform METHOD=POST
  ACTION="/PKIServ/clientauth-cgi/camodify.rexx">
<INPUT NAME="action" TYPE="hidden" VALUE="revoke">
<INPUT TYPE="submit" VALUE="Revoke">

```

```

<SELECT NAME="reason">
  <OPTION Selected VALUE="0">No Reason
  <OPTION VALUE="1">User key was compromised
  <OPTION VALUE="2">CA key was compromised
  <OPTION VALUE="3">User changed affiliation
  <OPTION VALUE="4">Certificate was superseded
  <OPTION VALUE="5">Original use no longer valid
</SELECT>
</ul>
</FORM>
<p>
<FORM METHOD=GET ACTION="/PKIServ/public-cgi/camain.rexx">
<center>
<INPUT TYPE="submit" VALUE="Home Page">
</FORM>
</center>
<p> %%-pagefooter%%
</BODY>
</HTML>
</RECONTENT>
<RESUCCESSCONTENT>
  %%-renewrevokeok%%
</RESUCCESSCONTENT>
<REFAILURECONTENT>
  %%-renewrevokebad%%
</REFAILURECONTENT>
<ADMINHEADER>
<HTML><HEAD>
<TITLE> Web Based Certificate Generation Administration </TITLE>
%%-copyright%%
</HEAD>
<BODY>
</ADMINHEADER>
<ADMINFOOTER>
<p> %%-pagefooter%%
</BODY>
</HTML>
</ADMINFOOTER>
</APPLICATION>

```

```

#
# =====
#
# Sample Templates - Browser and Server Certificate Requesting
#
# =====
#
# Template Name - 1 Year SAF Server Certificate

```

```

#
# Function - Allows end users to request certificates for servers
# using native SAF certificate generation facilities. The end user
# may provide values for any of the following fields:
#
# CommonName - optional
# OrgUnit - required
# Org - required
# Locality - optional
# StateProv - optional
# Country - required
# AltEmail - optional
# AltDomain - optional
# AltURI - optional
# AltIPAddr - optional
# Label - required
# PublicKey - required (This is the PKCS#10 request)
#
# PKISERV will provide the authenticated client UserId. The certificate
# will be used for handshaking only (e.g., SSL) and is good for 1
# year. The CERTAUTH certificate with Label "Local SAF CA" will be
# used for signing the certificate
#
# =====
#
<TEMPLATE NAME=1 Year SAF Server Certificate>
<TEMPLATE NAME=SAF Server Certificate>
<CONTENT>
<HTML><HEAD>
<TITLE> Web Based SAF Certificate Generation Application Pg 2</TITLE>
%%-copyright%%
<SCRIPT LANGUAGE="JavaScript">
<!--
function MissingRequiredFAlert(){
var STRING_MissingRequiredFPrompt=
    "Enter the field value that's not optional."

if ((document.serverform.Label.value=="") ||
    (document.serverform.OrgUnit.value=="") ||
    (document.serverform.Org.value=="") ||
    (document.serverform.Country.value=="") ||
    (document.serverform.PublicKey.value==""))
{
    alert(STRING_MissingRequiredFPrompt);
    if (document.serverform.OrgUnit.value=="")
        document.serverform.OrgUnit.focus();
    else if (document.serverform.Org.value=="")
        document.serverform.Org.focus();
    else if (document.serverform.Country.value=="")

```



```

        document.serverform.Country.focus();
    else if (document.serverform.Label.value=="")
        document.serverform.Label.focus();
    else
        document.serverform.PublicKey.focus();

    return true;
}
else {
return false;
}
}
}
//-->
</SCRIPT>

</HEAD>
<BODY>
<H1> SAF Server Certificate 1 Year (Auto Approved)</H1>
<p>
<H2>Choose one of the following:</H2>
<p>
<ul>
<h3><li>Request a New Certificate</h3>
# This ACTION forces userid/pw authentication and runs the task under
# the client's ID
#<FORM NAME=serverform METHOD=POST ACTION=
#          "/PKIServ/ssl-cgi-bin/auth/careq.rexx" onSubmit=

# This ACTION forces userid/pw authentication but runs the task under
# the surrogate ID
<FORM NAME=serverform METHOD=POST ACTION=
          "/PKIServ/ssl-cgi-bin/surrogateauth/careq.rexx" onSubmit=

# This ACTION is for non z/OS clients. The task runs under the
# surrogate ID
#<FORM NAME=serverform METHOD=POST ACTION=
#          "/PKIServ/ssl-cgi-bin/careq.rexx" onSubmit=
          "if(MissingRequiredFAlert()) return false; else return true;">

<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
<p> Enter values for the following field(s)
%%CommonName (Optional)%%
%%OrgUnit%%
%%OrgUnit2 (Optional)%%
%%Org%%
%%Locality (Optional)%%
%%StateProv (Optional)%%
%%Country%%
%%AltEmail (Optional)%%

```

```

%%AltDomain (Optional)%%
%%AltURI (Optional)%%
%%AltIPAddr (Optional)%%
%%Label%%
%%PublicKey%%
<p>
<INPUT TYPE="submit" VALUE="Submit certificate request"
ONCLICK="if(MissingRequiredFAlert()) return false; else return true;">
<INPUT TYPE="reset" VALUE="Clear">
</FORM>
<p>
<H3><li>Pick Up a Previously Issued Certificate</H3>

<FORM METHOD=GET ACTION="/PKIServ/ssl-cgi/caretrieve.rexx">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
<INPUT TYPE="submit" VALUE="Retrieve your certificate">
</FORM>
</ul>
<p>%%-pagefooter%%
</BODY>
</HTML>
</CONTENT>
<APPL>
  %%UserId%%
</APPL>
<CONSTANT>
  %%KeyUsage=handshake%%
  %%NotAfter=365%%
  %%SignWith=SAF:CERTAUTH/taca%%
</CONSTANT>
<SUCCESSCONTENT>
  %%-requestok%%
</SUCCESSCONTENT>
<FAILURECONTENT>
  %%-requestbad%%
</FAILURECONTENT>

<RETRIEVECONTENT>
<HTML><HEAD>
<TITLE> Web Based SAF Certificate Generation Application Pg 3</TITLE>
%%-copyright%%
<SCRIPT LANGUAGE="JavaScript">
<!--
function MissingTransIdAlert(){
var STRING_MissingTransIdPrompt=
  "Enter the transaction ID assigned to the certificate.";
if(document.retrieveform.TransactionId.value==""){
  alert(STRING_MissingTransIdPrompt);
  document.retrieveform.TransactionId.focus();

```

```

    return true;
}
else {
    return false;
}
}
//-->
</SCRIPT>
</HEAD>

<BODY>
<H1> Retrieve Your [tmplname]</H1>
<H3>Please bookmark this page</h3>
<p>Since your certificate may not have been issued yet, we recommend
that you create a bookmark to this location so that when you return to
this bookmark, the browser will display your transaction ID.
This is the easiest way to check your status.

# This ACTION forces userid/pw authentication and runs the task
# under the client's ID
<FORM NAME=retrieveform METHOD=GET ACTION=
    "/PKIServ/ssl-cgi/auth/cagetcert.rexx" onSubmit=
#
# This ACTION forces userid/pw authentication but runs the task
# under the surrogate ID
#<FORM NAME=retrieveform METHOD=GET ACTION=
#    "/PKIServ/ssl-cgi/surrogateauth/cagetcert.rexx" onSubmit=
#
# This ACTION is for non z/OS clients. The task runs under surrogate ID
#<FORM NAME=retrieveform METHOD=GET ACTION=
#    "/PKIServ/ssl-cgi/cagetcert.rexx" onSubmit=
#        "if(MissingTransIdAlert()) return false; else return true;">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
<p> Enter values for the following field(s)
    %%TransactionId%%
<p>
<INPUT TYPE="submit" VALUE="Continue">
</FORM>
<p>%%-pagefooter%%
</BODY>
</HTML>
</RETRIEVECONTENT>
<RETURNCERT>
%%-returnpkcs10cert%%
</RETURNCERT>
</TEMPLATE>

#
# =====

```

```

#
# Template Name - 1 Year SAF Browser Certificate
#
# Function - Allows end users to request certificates for their
# browsers using native SAF certificate generation facilities. The end
# user may provide values for any of the following fields:
#
# Label - required
# PublicKey - required (Provided by the browser itself)
#
# PKISERV will provide the authenticated client UserId. The certificate
# will be used for handshaking only (e.g., SSL) and is good for 1
# year. The CERTAUTH certificate with Label "Local SAF CA" will be
# used for signing the certificate. The Subject's Distinguished Name
# will be formed as:
#
# C=US/O=The Firm/OU=SAF template certificate/
#     OU=Nuts and Bolts Division/CN=<determined by SAF>
#
# The presence of CommonName without a value tells SAF to determine
# the CN value from the PGMNAME field of the user's USER profile.
# See the RACF Callable Services Guide for more information
#
# =====
#
<TEMPLATE NAME=1 Year SAF Browser Certificate>
<TEMPLATE NAME=SAF Browser Certificate>
<CONTENT>
<HTML><HEAD>
<TITLE> Web Based SAF Certificate Generation Application Pg 2</TITLE>
%%-copyright%%
%%-AdditionalHead[browsertype]%%
<SCRIPT LANGUAGE="JavaScript">
<!--
function MissingRequiredFAlert(){
var STRING_MissingRequiredFPrompt=
    "Enter the field that's required."
if(document.CertReq.Label.value==""){
alert(STRING_MissingRequiredFPrompt);
document.CertReq.Label.focus();
return true;
}
else {
return false;
}
}
//-->
</SCRIPT>
</HEAD>

```

```

<BODY>
<H1> SAF Browser Certificate 1 Year (Auto Approved)</H1>
<p>
<H2>Choose one of the following:</H2>
<p>
<ul>
<h3><li>Request a New Certificate</h3>
# This ACTION forces userid/pw authentication and runs the task under
# the client's ID
#<FORM NAME="CertReq" METHOD=POST ACTION=
#      "/PKIServ/ssl-cgi-bin/auth/careq.rexx" onSubmit=

# This ACTION forces userid/pw authentication but runs the task under
# the surrogate ID
  <FORM NAME="CertReq" METHOD=POST ACTION=
      "/PKIServ/ssl-cgi-bin/surrogateauth/careq.rexx" onSubmit=

# This ACTION is for non z/OS clients. The task runs under the
# surrogate ID
#<FORM NAME="CertReq" METHOD=POST ACTION=
#      "/PKIServ/ssl-cgi-bin/careq.rexx" onSubmit=
#      "if(MissingRequiredFAlert()) return false; else return true;">

<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
<p> Enter values for the following field(s)
  %%Label%%
  %%PublicKey[browsertype]%%
<INPUT TYPE="reset" VALUE="Clear">
</FORM>
<p>
<H3><li>Pick Up a Previously Issued Certificate</H3>
<FORM METHOD=GET ACTION="/PKIServ/ssl-cgi/caretrieve.rexx">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
<INPUT TYPE="submit" VALUE="Retrieve your certificate">
</FORM>
</ul>
<p>%%-pagefooter%%
</BODY>
</HTML>
</CONTENT>
<APPL>
  %%UserId%%
</APPL>
<CONSTANT>
  %%KeyUsage=handshake%%
  %%NotAfter=365%%
  %%OrgUnit=SAF template certificate%%
  %%OrgUnit=Nuts and Bolts Division%%

```

```

%%Org=The Firm%%
%%Country=US%%
%%SignWith=SAF:CERTAUTH/taca%%
%%CommonName=%%
</CONSTANT>
<SUCCESSCONTENT>
  %%-requestok%%
</SUCCESSCONTENT>
<FAILURECONTENT>
  %%-requestbad%%
</FAILURECONTENT>
<RETRIEVECONTENT>

<HTML><HEAD>
%%-copyright%%
<TITLE> Web Based SAF Certificate Generation Application Pg 3</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
function MissingTransIdAlert(){
var STRING_MissingTransIdPrompt=
  "Enter the transaction ID assigned to the certificate.";
if(document.retrieveform.TransactionId.value==""){
  alert(STRING_MissingTransIdPrompt);
  document.retrieveform.TransactionId.focus();
  return true;
}
else {
  return false;
}
}
//-->
</SCRIPT>
</HEAD>

<BODY>
<H1> Retrieve Your [tmplname]</H1>
<H3>Please bookmark this page</h3>
<p>Since your certificate may not have been issued yet, we recommend
that you create a bookmark to this location so that when you return to
this bookmark, the browser will display your transaction ID.
This is the easiest way to check your status.

# This ACTION forces userid/pw authentication and runs the task
# under the client's ID
<FORM NAME=retrieveform METHOD=GET ACTION=
  "/PKIServ/ssl-cgi/auth/cagetcert.rexx" onSubmit=
#
# This ACTION forces userid/pw authentication but runs the task
# under the surrogate ID

```

```

#<FORM NAME=retrieveform METHOD=GET ACTION=
#      "/PKIServ/ssl-cgi/surrogateauth/cagetcert.rexx" onSubmit=
#
# This ACTION is for non z/OS clients. The task runs under surrogate ID
#<FORM NAME=retrieveform METHOD=GET ACTION=
#      "/PKIServ/ssl-cgi/cagetcert.rexx" onSubmit=
#          "if(MissingTransIdAlert()) return false; else return true;">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
    %%TransactionId%%
<p>
<INPUT TYPE="submit" VALUE="Retrieve and Install Certificate">
</FORM>
<p>
<H2>To check that your certificate installed properly, follow the
procedure below:</h2>
<p><B>Netscape V6</B> - Click Edit->Preferences, then Privacy and Security->
Certificates. Click the Manage Certificates button to start the Certificate Manager.
Your new certificate should appear in the Your Certificates list.
Select it then click View to see more information.
<p><B>Netscape V4</B> - Click the Security button, then Certificates->
Yours. Your certificate should appear in the list. Select it then
click Verify.
<p><B>Internet Explorer V5</B> - Click Tools->Internet Options, then
Content, Certificates.
Your certificate should appear in the Personal list. Click Advanced to
see additional information.
<p>
<FORM METHOD=GET ACTION="/PKIServ/public-cgi/camain.rexx">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
<INPUT TYPE="submit" VALUE="Home page">
</FORM>
<p>%%-pagefooter%%
</BODY>
</HTML>
</RETRIEVECONTENT>
<RETURNCERT>
%%returnbrowsercert[browsertype]%%
</RETURNCERT>
</TEMPLATE>

#
#
# =====
#
# Template Name - 1 Year PKI SSL Browser Certificate
#
# Function - Creates a 1 year certificate good for general SSL client
#            authentication using a browser. If approved, the
#            certificate becomes valid after it's requested.

```

```

#           (You may delay the valid date by specifying a non zero
#           number for the value of 'NotBefore',
#           eg. NotBefore=5. That means if the request is approved,
#           the certificate will become valid 5 days after it's
#           requested.)
#
#           These certificates will be stored in LDAP if The O= and
#           OU= suffixes have already been created
#
# Other than the user input fields, all other information is hard coded.
#
# User input fields:
#   CommonName - required
#   Requestor - optional
#   PassPhrase - required
#   PublicKey - required (Provided by the browser itself)
#   NotifyEmail - optional
#   Email - optional
#
#   RACF userid/password authentication : not require
#   Administrator approval             : require
#
# =====
#
<TEMPLATE NAME=1 Year PKI SSL Browser Certificate>
<TEMPLATE NAME=PKI Browser Certificate>
<NICKNAME=1YBSSL>
<CONTENT>
<HTML><HEAD>
<TITLE> Web Based PKIX Certificate Generation Application Pg 2</TITLE>
%%-copyright%%
%%-AdditionalHead[browsertype]%%
<SCRIPT LANGUAGE="JavaScript">
<!--
function ValidateEntry(){
var STRING_MissingFieldPrompt=
    "Enter the required field."
var STRING_MissingConfirmPwdPrompt=
    "Reenter password."
var STRING_UnmatchPwdPrompt=
    "The passwords do not match. Enter again."
var STRING_UnmatchEmailPrompt=
    "The Email addresses for Distinguished name and notification do not match.
Enter again."
if(document.CertReq.CommonName.value=="") {
alert(STRING_MissingFieldPrompt);
document.CertReq.CommonName.focus();
return true;
}
}

```



```

else if(document.CertReq.PassPhrase.value=="") {
alert(String_MissingFieldPrompt);
document.CertReq.PassPhrase.focus();
return true;
}
else if(document.CertReq.ConfirmPassPhrase.value=="") {
alert(String_MissingConfirmPwdPrompt);
document.CertReq.ConfirmPassPhrase.focus();
return true;
}
else if(document.CertReq.PassPhrase.value!=
document.CertReq.ConfirmPassPhrase.value){
alert(String_UnmatchPwdPrompt);
document.CertReq.ConfirmPassPhrase.focus();
return true;
}
else if((document.CertReq.Email.value !="" &&
document.CertReq.NotifyEmail.value != "") &&
(document.CertReq.Email.value!=
document.CertReq.NotifyEmail.value)){
alert(String_UnmatchEmailPrompt);
document.CertReq.NotifyEmail.focus();
return true;
}
else {
return false;
}
}
//-->
</SCRIPT>
</HEAD>

<BODY>
<H1>1 Year SSL Browser Certificate</H1>
<p>
<H2>Choose one of the following:</H2>
<p>
<ul>
<h3><li>Request a New Certificate</h3>
# This ACTION forces userid/pw authentication and runs the task under
# the client's ID
#<FORM NAME="CertReq" METHOD=POST ACTION=
# "/PKIServ/ssl-cgi-bin/auth/careq.rexx" onSubmit=

# This ACTION forces userid/pw authentication but runs the task under
# the surrogate ID
#<FORM NAME="CertReq" METHOD=POST ACTION=
# "/PKIServ/ssl-cgi-bin/surrogateauth/careq.rexx" onSubmit=

```

```

# This ACTION is for non z/OS clients. The task runs under the
# surrogate ID
<FORM NAME="CertReq" METHOD=POST ACTION=
    "/PKIServ/ssl-cgi-bin/careq.rexx" onSubmit=
    "if(ValidateEntry()) return false; else return true;">

<INPUT NAME="Template" TYPE="hidden" VALUE="[tplname]">
<p> Enter values for the following field(s)
%%CommonName%%
%%Email (optional)%%
%%Requestor (optional)%%
%%NotifyEmail (optional)%%
%%PassPhrase%%
%%PublicKey2[browsertype]%%
<INPUT TYPE="reset" VALUE="Clear">
</FORM>
<p>
<H3><li>Pick Up a Previously Issued Certificate</H3>
<FORM METHOD=GET ACTION="/PKIServ/ssl-cgi/caretrieve.rexx">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tplname]">
<INPUT TYPE="submit" VALUE="Retrieve your certificate">
</FORM>
</ul>
<p>%%-pagefooter%%
</BODY>
</HTML>
</CONTENT>
<CONSTANT>
%%NotBefore=0%%
%%NotAfter=365%%
%%KeyUsage=handshake%%
%%OrgUnit=Class 1 Internet Certificate CA%%
%%Org=The Firm%%
%%SignWith=PKI:%%
</CONSTANT>
<ADMINAPPROVE>
%%CommonName (Optional)%%
%%OrgUnit (Optional)%%
%%OrgUnit (Optional)%%
%%Org (Optional)%%
%%NotBefore (optional)%%
%%NotAfter (Optional)%%
%%KeyUsage (Optional)%%
%%HostIdMap (Optional)%%
%%HostIdMap (Optional)%%
%%HostIdMap (Optional)%%
%%HostIdMap (Optional)%%
</ADMINAPPROVE>
<SUCCESSCONTENT>

```

```

%%-requestok%%
</SUCCESSCONTENT>
<FAILURECONTENT>
%%-requestbad%%
</FAILURECONTENT>

<RETRIEVECONTENT>
<HTML><HEAD>
%%-copyright%%
<TITLE> Web Based PKIX Certificate Generation Application Pg 3</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
function MissingTransIdAlert(){
var STRING_MissingTransIdPrompt=
    "Enter the transaction ID assigned to the certificate.";
if(document.retrieveform.TransactionId.value==""){
    alert(STRING_MissingTransIdPrompt);
    document.retrieveform.TransactionId.focus();
    return true;
}
else {
    return false;
}
}
//-->
</SCRIPT>
</HEAD>

<BODY>
<H1> Retrieve Your [tplname]</H1>
<H3>Please bookmark this page</h3>
<p>Since your certificate may not have been issued yet, we recommend
that you create a bookmark to this location so that when you return to
this bookmark, the browser will display your transaction ID.
This is the easiest way to check your status.

# This ACTION forces userid/pw authentication and runs the task
# under the client's ID
#<FORM NAME=retrieveform METHOD=POST ACTION=
#     "/PKIServ/ssl-cgi-bin/auth/cagetcert.rexx" onSubmit=
#
# This ACTION forces userid/pw authentication but runs the task
# under the surrogate ID
#<FORM NAME=retrieveform METHOD=POST ACTION=
#     "/PKIServ/ssl-cgi-bin/surrogateauth/cagetcert.rexx" onSubmit=
#
# This ACTION is for non z/OS clients. The task runs under surrogate ID
<FORM NAME=retrieveform METHOD=POST ACTION=
    "/PKIServ/ssl-cgi-bin/cagetcert.rexx" onSubmit=

```

```

        "if(MissingTransIdAlert()) return false; else return true;">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
    %%TransactionId%%
    %%ChallengePassPhrase (optional)%%
<p>
<INPUT TYPE="submit" VALUE="Retrieve and Install Certificate">
</FORM>
<p>
<H2>To check that your certificate installed properly, follow the
procedure below:</h2>
<p><B>Netscape V6</B> - Click Edit->Preferences, then Privacy and Security->
Certificates. Click the Manage Certificates button to start the Certificate Manager.
Your new certificate should appear in the Your Certificates list.
Select it then click View to see more information.
<p><B>Netscape V4</B> - Click the Security button, then Certificates->
Yours. Your certificate should appear in the list. Select it then
click Verify.
<p><B>Internet Explorer V5</B> - Click Tools->Internet Options, then
Content, Certificates.
Your certificate should appear in the Personal list. Click Advanced to
see additional information.
<p>
<FORM METHOD=GET ACTION="/PKIServ/public-cgi/camain.rexx">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
<INPUT TYPE="submit" VALUE="Home page">
</FORM>
<p>%%-pagefooter%%
</BODY>
</HTML>
</RETRIEVECONTENT>
<RETURNCERT>
%%returnbrowsercert[browsertype]%%
</RETURNCERT>
</TEMPLATE>

#
#
# =====
#
# Template Name - 1 Year PKI S/MIME Browser Certificate
#
# Function - Creates a 1 year certificate good for S/MIME
#             authentication using a browser. If approved, the
#             certificate becomes valid after it's requested.
#             (You may delay the valid date by specifying a non zero
#             number for the value of 'NotBefore',
#             eg. NotBefore=5. That means if the request is approved,
#             the certificate will become valid 5 days after it's
#             requested.)

```

```

#
#           These certificates will be stored in LDAP if The O= and
#           OU= suffixes have already been created
#
# Other than the user input fields, all other information is hard coded.
#
# User input fields:
#   CommonName - required
#   AltEmail - required
#   Requestor - optional
#   PassPhrase - required
#   PublicKey - required (Provided by the browser itself)
#   NotifyEmail- optional
#
#   RACF userid/password authentication : not required
#   Administrator approval             : required
#
#
# =====
#
<TEMPLATE NAME=1 Year PKI S/MIME Browser Certificate>
<TEMPLATE NAME=PKI Browser Certificate>
<NICKNAME=1YBSM>
<CONTENT>
<HTML><HEAD>
<TITLE> Web Based PKIX Certificate Generation Application Pg 2</TITLE>
%%-copyright%%
%%-AdditionalHead[browsertype]%%
<SCRIPT LANGUAGE="JavaScript">
<!--
function ValidateEntry(){
var STRING_MissingFieldPrompt=
    "Enter the required field."
var STRING_MissingConfirmPwdPrompt=
    "Reenter password."
var STRING_MissingPwdPrompt=
    "Need to enter password before confirm it."
var STRING_UnmatchPwdPrompt=
    "The passwords do not match. Enter again."
if(document.CertReq.CommonName.value=="") {
alert(STRING_MissingFieldPrompt);
document.CertReq.CommonName.focus();
return true;
}
if(document.CertReq.AltEmail.value=="") {
alert(STRING_MissingFieldPrompt);
document.CertReq.AltEmail.focus();
return true;
}
}

```

```

else if(document.CertReq.PassPhrase.value=="") {
alert(String_MissingFieldPrompt);
document.CertReq.PassPhrase.focus();
return true;
}
else if(document.CertReq.ConfirmPassPhrase.value=="") {
alert(String_MissingConfirmPwdPrompt);
document.CertReq.ConfirmPassPhrase.focus();
return true;
}
else if(document.CertReq.PassPhrase.value!=
document.CertReq.ConfirmPassPhrase.value){
alert(String_UnmatchPwdPrompt);
document.CertReq.ConfirmPassPhrase.focus();
return true;
}
else {
return false;
}
}
//-->
</SCRIPT>
</HEAD>

<BODY>
<H1>1 Year S/MIME Browser Certificate</H1>
<p>
<H2>Choose one of the following:</H2>
<p>
<ul>
<h3><li>Request a New Certificate</h3>
# This ACTION forces userid/pw authentication and runs the task under
# the client's ID
#<FORM NAME="CertReq" METHOD=POST ACTION=
# "/PKIServ/ssl-cgi-bin/auth/careq.rexx" onSubmit=

# This ACTION forces userid/pw authentication but runs the task under
# the surrogate ID
#<FORM NAME="CertReq" METHOD=POST ACTION=
# "/PKIServ/ssl-cgi-bin/surrogateauth/careq.rexx" onSubmit=

# This ACTION is for non z/OS clients. The task runs under the
# surrogate ID
<FORM NAME="CertReq" METHOD=POST ACTION=
"/PKIServ/ssl-cgi-bin/careq.rexx" onSubmit=
"if(ValidateEntry()) return false; else return true;">

<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
<p> Enter values for the following field(s)

```

```

%%CommonName%%
%%AltEmail%%
%%Requestor (optional)%%
%%NotifyEmail (optional)%%
%%PassPhrase%%
%%PublicKey2[browsertype]%%
<INPUT TYPE="reset" VALUE="Clear">
</FORM>
<p>
<H3><li>Pick Up a Previously Issued Certificate</H3>
<FORM METHOD=GET ACTION="/PKIServ/ssl-cgi/caretrieve.rexx">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tplname]">
<INPUT TYPE="submit" VALUE="Retrieve your certificate">
</FORM>
</ul>
<p>%%-pagefooter%%
</BODY>
</HTML>
</CONTENT>
<CONSTANT>
%%NotBefore=0%%
%%NotAfter=365%%
%%KeyUsage=handshake%%
%%OrgUnit=Class 1 Internet Certificate CA%%
%%Org=The Firm%%
%%SignWith=PKI:%%
</CONSTANT>
<ADMINAPPROVE>
%%CommonName (Optional)%%
%%OrgUnit (Optional)%%
%%Org (Optional)%%
%%AltEmail (optional)%%
%%NotBefore (optional)%%
%%NotAfter (Optional)%%
%%KeyUsage (Optional)%%
</ADMINAPPROVE>
<SUCCESSCONTENT>
%%-requestok%%
</SUCCESSCONTENT>
<FAILURECONTENT>
%%-requestbad%%
</FAILURECONTENT>

<RETRIEVECONTENT>
<HTML><HEAD>
%%-copyright%%
<TITLE> Web Based PKIX Certificate Generation Application Pg 3</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--

```

```

function MissingTransIdAlert(){
var STRING_MissingTransIdPrompt=
    "Enter the transaction ID assigned to the certificate.";
if(document.retrieveform.TransactionId.value==""){
    alert(STRING_MissingTransIdPrompt);
    document.retrieveform.TransactionId.focus();
    return true;
}
else {
    return false;
}
}
//-->
</SCRIPT>
</HEAD>

<BODY>
<H1> Retrieve Your [tmplname]</H1>
<H3>Please bookmark this page</h3>
<p>Since your certificate may not have been issued yet, we recommend
that you create a bookmark to this location so that when you return to
this bookmark, the browser will display your transaction ID.
This is the easiest way to check your status.

# This ACTION forces userid/pw authentication and runs the task
# under the client's ID
#<FORM NAME=retrieveform METHOD=POST ACTION=
#    "/PKIServ/ssl-cgi-bin/auth/cagetcert.rexx" onSubmit=
#
# This ACTION forces userid/pw authentication but runs the task
# under the surrogate ID
#<FORM NAME=retrieveform METHOD=POST ACTION=
#    "/PKIServ/ssl-cgi-bin/surrogateauth/cagetcert.rexx" onSubmit=
#
# This ACTION is for non z/OS clients. The task runs under surrogate ID
<FORM NAME=retrieveform METHOD=POST ACTION=
    "/PKIServ/ssl-cgi-bin/cagetcert.rexx" onSubmit=
    "if(MissingTransIdAlert()) return false; else return true;">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
    %%TransactionId%%
    %%ChallengePassPhrase (optional)%%
<p>
<INPUT TYPE="submit" VALUE="Retrieve and Install Certificate">
</FORM>
<p>
<H2>To check that your certificate installed properly, follow the
procedure below:</h2>
<p><B>Netscape V6</B> - Click Edit->Preferences, then Privacy and Security->
Certificates. Click the Manage Certificates button to start the Certificate Manager.

```


Your new certificate should appear in the Your Certificates list. Select it then click View to see more information.

<p>Netscape V4 - Click the Security button, then Certificates->Yours. Your certificate should appear in the list. Select it then click Verify.

<p>Internet Explorer V5 - Click Tools->Internet Options, then Content, Certificates.

Your certificate should appear in the Personal list. Click Advanced to see additional information.

```
<p>
<FORM METHOD=GET ACTION="/PKIServ/public-cgi/camain.rexx">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
<INPUT TYPE="submit" VALUE="Home page">
</FORM>
<p>%%-pagefooter%%
</BODY>
</HTML>
</RETRIEVECONTENT>
<RETURNCERT>
%%returnbrowsercert[browsertype]%%
</RETURNCERT>
</TEMPLATE>
#
#
# =====
#
# Template Name - 2 Year PKI Browser Certificate For Authenticating
#                 to z/OS
#
# Function - Creates a 2 year certificate good for authenticating to
#            z/OS. If approved, the certificate becomes valid after
#            it's requested.
#            (You may delay the valid date by specifying a non zero
#            number for the value of 'NotBefore',
#            eg. NotBefore=5. That means if the request is approved,
#            the certificate will become valid 5 days after it's
#            requested.)
#            HostidMap is formed by putting %%Userid%% and
#            %%HostIdMap=@host-name in the APPL section.
#
#            These certificates will be stored in LDAP if The 0= and
#            OU= suffixes have already been created
#
# Other than the user input fields, all other information is hard coded.
#
# User input fields:
# Requestor - optional
# PassPhrase - required
# PublicKey - required (Provided by the browser itself)
```

```

# NotifyEmail - optional
#
# The presence of CommonName without a value tells SAF to determine
# the CN value from the PGMNAME field of the user's USER profile.
# See the RACF Callable Services Guide for more information
#
# RACF userid/password authentication : require
# Administrator approval             : not require
#
#
# =====
#
<TEMPLATE NAME=2 Year PKI Browser Certificate For Authenticating To z/OS>
<TEMPLATE NAME=PKI Browser Certificate>
<NICKNAME=2YBZOS>
<CONTENT>
<HTML><HEAD>
<TITLE> Web Based PKIX Certificate Generation Application Pg 2</TITLE>
%%-copyright%%
%%-AdditionalHead[browsertype]%%
<SCRIPT LANGUAGE="JavaScript">
<!--
function ValidateEntry(){
var STRING_MissingFieldPrompt=
        "Enter the required field."
var STRING_MissingConfirmPwdPrompt=
        "Reenter password."
var STRING_MissingPwdPrompt=
        "Need to enter password before confirm it."
var STRING_UnmatchPwdPrompt=
        "The passwords do not match. Enter again."
if(document.CertReq.PassPhrase.value=="") {
alert(STRING_MissingFieldPrompt);
document.CertReq.PassPhrase.focus();
return true;
}
else if(document.CertReq.ConfirmPassPhrase.value=="") {
alert(STRING_MissingConfirmPwdPrompt);
document.CertReq.ConfirmPassPhrase.focus();
return true;
}
else if(document.CertReq.PassPhrase.value!=
        document.CertReq.ConfirmPassPhrase.value){
alert(STRING_UnmatchPwdPrompt);
document.CertReq.ConfirmPassPhrase.focus();
return true;
}
else {
return false;
}
}

```

```

}
}
//-->
</SCRIPT>
</HEAD>

<BODY>
<H1>2 Year Browser Certificate For Authenticating To z/OS</H1>
<p>
<H2>Choose one of the following:</H2>
<p>
<ul>
<h3><li>Request a New Certificate</h3>
# This ACTION forces userid/pw authentication and runs the task under
# the client's ID
#<FORM NAME="CertReq" METHOD=POST ACTION=
#           "/PKIServ/ssl-cgi-bin/auth/careq.rexx" onSubmit=

# This ACTION forces userid/pw authentication but runs the task under
# the surrogate ID
<FORM NAME="CertReq" METHOD=POST ACTION=
           "/PKIServ/ssl-cgi-bin/surrogateauth/careq.rexx" onSubmit=

# This ACTION is for non z/OS clients. The task runs under the
# surrogate ID
#<FORM NAME="CertReq" METHOD=POST ACTION=
#           "/PKIServ/ssl-cgi-bin/careq.rexx" onSubmit=
#           "if(ValidateEntry()) return false; else return true;">

<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
<p> Enter values for the following field(s)
%%Requestor (optional)%%
%%NotifyEmail (optional)%%
%%PassPhrase%%
%%PublicKey2[browsertype]%%
<INPUT TYPE="reset" VALUE="Clear">
</FORM>
<p>
<H3><li>Pick Up a Previously Issued Certificate</H3>
<FORM METHOD=GET ACTION="/PKIServ/ssl-cgi/caretrieve.rexx">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
<INPUT TYPE="submit" VALUE="Retrieve your certificate">
</FORM>
</ul>
<p>%%-pagefooter%%
</BODY>
</HTML>
</CONTENT>
<APPL>

```

```

%%UserId%%
%%HostIdMap=@host-name%%
</APPL>
<CONSTANT>
%%NotBefore=0%%
%%NotAfter=730%%
%%KeyUsage=handshake%%
%%OrgUnit=Class 1 Internet Certificate CA%%
%%Org=The Firm%%
%%SignWith=PKI:%%
%%CommonName=%%
</CONSTANT>
<SUCCESSCONTENT>
%%-requestok%%
</SUCCESSCONTENT>
<FAILURECONTENT>
%%-requestbad%%
</FAILURECONTENT>

<RETRIEVECONTENT>
<HTML><HEAD>
%%-copyright%%
<TITLE> Web Based PKIX Certificate Generation Application Pg 3</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
function MissingTransIdAlert(){
var STRING_MissingTransIdPrompt=
    "Enter the transaction ID assigned to the certificate.";
if(document.retrieveform.TransactionId.value==""){
    alert(STRING_MissingTransIdPrompt);
    document.retrieveform.TransactionId.focus();
    return true;
}
else {
    return false;
}
}
//-->
</SCRIPT>
</HEAD>

<BODY>
<H1> Retrieve Your [tplname]</H1>
<H3>Please bookmark this page</h3>
<p>Since your certificate may not have been issued yet, we recommend
that you create a bookmark to this location so that when you return to
this bookmark, the browser will display your transaction ID.
This is the easiest way to check your status.

```

```

# This ACTION forces userid/pw authentication and runs the task
# under the client's ID
#<FORM NAME=retrieveform METHOD=POST ACTION=
#      "/PKIServ/ssl-cgi-bin/auth/cagetcert.rexx" onSubmit=
#
# This ACTION forces userid/pw authentication but runs the task
# under the surrogate ID
<FORM NAME=retrieveform METHOD=POST ACTION=
      "/PKIServ/ssl-cgi-bin/surrogateauth/cagetcert.rexx" onSubmit=
#
# This ACTION is for non z/OS clients. The task runs under surrogate ID
#<FORM NAME=retrieveform METHOD=POST ACTION=
#      "/PKIServ/ssl-cgi-bin/cagetcert.rexx" onSubmit=
#      "if(MissingTransIdAlert()) return false; else return true;">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
%%TransactionId%%
%%ChallengePassPhrase (optional)%%
<p>
<INPUT TYPE="submit" VALUE="Retrieve and Install Certificate">
</FORM>
<p>
<H2>To check that your certificate installed properly, follow the
procedure below:</h2>
<p><B>Netscape V6</B> - Click Edit->Preferences, then Privacy and Security->
Certificates. Click the Manage Certificates button to start the Certificate Manager.
Your new certificate should appear in the Your Certificates list.
Select it then click View to see more information.
<p><B>Netscape V4</B> - Click the Security button, then Certificates->
Yours. Your certificate should appear in the list. Select it then
click Verify.
<p><B>Internet Explorer V5</B> - Click Tools->Internet Options, then
Content, Certificates.
Your certificate should appear in the Personal list. Click Advanced to
see additional information.
<p>
<FORM METHOD=GET ACTION="/PKIServ/public-cgi/camain.rexx">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
<INPUT TYPE="submit" VALUE="Home page">
</FORM>
<p>%%-pagefooter%%
</BODY>
</HTML>
</RETRIEVECONTENT>
<RETURNCERT>
%%returnbrowsercert[browsertype]%%
</RETURNCERT>
</TEMPLATE>
#
#

```

```

# =====
#
# Template Name - 5 Year PKI SSL Server Certificate
#
# Function - Creates a 5 year Server certificate. If approved, the
#            certificate becomes valid after it's requested.
#            (You may delay the valid date by specifying a non zero
#            number for the value of 'NotBefore',
#            eg. NotBefore=5. That means if the request is approved,
#            the certificate will become valid 5 days after it's
#            requested.)
#
#            These certificates will be stored in LDAP if The O= and
#            OU= suffixes have already been created
#
# Other than the user input fields, all other information is hard coded.
#
# User input fields:
#   Email - optional
#   CommonName - optional
#   OrgUnit - optional
#   Org - optional
#   Street - optional
#   Locality - optional
#   StateProv - optional
#   PostalCode - optional
#   Country - optional
#   AltEmail - optional
#   AltDomain - optional
#   AltURI - optional
#   AltIPAddr - optional
#   PassPhrase - required
#   PublicKey - required (This is the PKCS#10 request)
#   NotifyEmail - optional
#
# RACF userid/password authentication : not require
# Administrator approval : require
#
# =====
#
<TEMPLATE NAME=5 Year PKI SSL Server Certificate>

<TEMPLATE NAME=PKI Server Certificate>

<NICKNAME=5YSSL>
<CONTENT>
<HTML><HEAD>
<TITLE> Web Based PKIX Certificate Generation Application Pg 2</TITLE>

```

```

%%-copyright%%
<SCRIPT LANGUAGE="JavaScript">
<!--
function MissingRequiredFAlert(){
var STRING_MissingRequiredFPrompt=
    "Enter the field value that's not optional."
var STRING_MissingConfirmPwdPrompt=
    "Reenter password."
var STRING_MissingPwdPrompt=
    "Need to enter password before confirm it."
var STRING_UnmatchPwdPrompt=
    "The passwords do not match. Enter again."
var STRING_UnmatchEmailPrompt=
    "The Email addresses for Distinguished name and notification do not match.
Enter again."

if (document.serverform.PassPhrase.value=="")
{
    alert(STRING_MissingRequiredFPrompt);
    document.serverform.PassPhrase.focus();
    return true;
}
else if(document.serverform.ConfirmPassPhrase.value=="") {
alert(STRING_MissingConfirmPwdPrompt);
document.serverform.ConfirmPassPhrase.focus();
return true;
}
else if(document.serverform.PassPhrase.value!=
    document.serverform.ConfirmPassPhrase.value){
alert(STRING_UnmatchPwdPrompt);
document.serverform.ConfirmPassPhrase.focus();
return true;
}
else if(document.serverform.PublicKey.value=="") {
alert(STRING_MissingRequiredFPrompt);
document.serverform.PublicKey.focus();
return true;
}
else if((document.serverform.Email.value != "" &&
    document.serverform.NotifyEmail.value != "" ) &&
    (document.serverform.Email.value!=
    document.serverform.NotifyEmail.value)){
alert(STRING_UnmatchEmailPrompt);
document.serverform.NotifyEmail.focus();
return true;
}
else {
return false;
}
}

```

```

}
//-->
</SCRIPT>

</HEAD>
<BODY>
<H1> 5 Year PKI SSL Server Certificate</H1>
<p>
<H2>Choose one of the following:</H2>
<p>
<ul>
<h3><li>Request a New Certificate</h3>
# This ACTION forces userid/pw authentication and runs the task under
# the client's ID
#<FORM NAME=serverform METHOD=POST ACTION=
#          "/PKIServ/ssl-cgi-bin/auth/careq.rexx" onSubmit=

# This ACTION forces userid/pw authentication but runs the task under
# the surrogate ID
#<FORM NAME=serverform METHOD=POST ACTION=
#          "/PKIServ/ssl-cgi-bin/surrogateauth/careq.rexx" onSubmit=

# This ACTION is for non z/OS clients. The task runs under the
# surrogate ID
<FORM NAME=serverform METHOD=POST ACTION=
          "/PKIServ/ssl-cgi-bin/careq.rexx" onSubmit=
          "if(MissingRequiredFAlert()) return false; else return true;">

<INPUT NAME="Template" TYPE="hidden" VALUE="[tplname]">
<p> Enter values for the following field(s)
%%Email (Optional)%%
%%CommonName (Optional)%%
%%OrgUnit (Optional)%%
%%OrgUnit2 (Optional)%%
%%Org (Optional)%%
%%Street (Optional)%%
%%Locality (Optional)%%
%%StateProv (Optional)%%
%%PostalCode (Optional)%%
%%Country (Optional)%%
%%AltEmail (Optional)%%
%%AltDomain (Optional)%%
%%AltURI (Optional)%%
%%AltIPAddr (Optional)%%
%%Requestor (Optional)%%
%%NotifyEmail (Optional)%%
%%PassPhrase%%
%%PublicKey%%
<p>

```



```

<INPUT TYPE="submit" VALUE="Submit certificate request"
ONCLICK="if(MissingRequiredFAlert()) return false; else return true;">
<INPUT TYPE="reset" VALUE="Clear">
</FORM>
<p>
<H3><li>Pick Up a Previously Issued Certificate</H3>

<FORM METHOD=GET ACTION="/PKIServ/ssl-cgi/caretrieve.rexx">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tplname]">
<INPUT TYPE="submit" VALUE="Retrieve your certificate">
</FORM>
</ul>
<p>%%-pagefooter%%
</BODY>
</HTML>
</CONTENT>
<CONSTANT>
%%NotBefore=0%%
%%NotAfter=1825%%
%%KeyUsage=handshake%%
%%SignWith=PKI:%%
</CONSTANT>
<ADMINAPPROVE>
%%CommonName (Optional)%%
%%OrgUnit (Optional)%%
%%OrgUnit (Optional)%%
%%Org (Optional)%%
%%Locality (Optional)%%
%%StateProv (Optional)%%
%%Country (Optional)%%
%%AltEmail (Optional)%%
%%AltDomain (Optional)%%
%%AltURI (Optional)%%
%%AltIPAddr (Optional)%%
%%NotBefore (optional)%%
%%NotAfter (Optional)%%
%%KeyUsage (Optional)%%
</ADMINAPPROVE>
<SUCCESSCONTENT>
%%-requestok%%
</SUCCESSCONTENT>
<FAILURECONTENT>
%%-requestbad%%
</FAILURECONTENT>

<RETRIEVECONTENT>
<HTML><HEAD>
<TITLE> Web Based PKIX Certificate Generation Application Pg 3</TITLE>
%%-copyright%%

```

```

<SCRIPT LANGUAGE="JavaScript">
<!--
function MissingTransIdAlert(){
var STRING_MissingTransIdPrompt=
    "Enter the transaction ID assigned to the certificate.";
if(document.retrieveform.TransactionId.value==""){
    alert(STRING_MissingTransIdPrompt);
    document.retrieveform.TransactionId.focus();
    return true;
}
else {
    return false;
}
}
//-->
</SCRIPT>
</HEAD>

<BODY>
<H1> Retrieve Your [tmplname]</H1>
<H3>Please bookmark this page</h3>
<p>Since your certificate may not have been issued yet, we recommend
that you create a bookmark to this location so that when you return to
this bookmark, the browser will display your transaction ID.
This is the easiest way to check your status.

# This ACTION forces userid/pw authentication and runs the task
# under the client's ID
#<FORM NAME=retrieveform METHOD=POST ACTION=
#    "/PKIServ/ssl-cgi-bin/auth/cagetcert.rexx" onSubmit=
#
# This ACTION forces userid/pw authentication but runs the task
# under the surrogate ID
#<FORM NAME=retrieveform METHOD=POST ACTION=
#    "/PKIServ/ssl-cgi-bin/surrogateauth/cagetcert.rexx" onSubmit=
#
# This ACTION is for non z/OS clients. The task runs under surrogate ID
<FORM NAME=retrieveform METHOD=POST ACTION=
    "/PKIServ/ssl-cgi-bin/cagetcert.rexx" onSubmit=
    "if(MissingTransIdAlert()) return false; else return true;">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
<p> Enter values for the following field(s)
    %%TransactionId%%
    %%ChallengePassPhrase (optional)%%
<p>
<INPUT TYPE="submit" VALUE="Continue">
</FORM>
<p>%%-pagefooter%%
</BODY>

```

```

</HTML>
</RETRIEVECONTENT>
<RETURNCERT>
%%-returnpkcs10cert%%
</RETURNCERT>
</TEMPLATE>

#
# =====
#
# Template Name - 5 Year PKI IPSEC Server (Firewall) Certificate
#
# Function - Creates a 5 year Server certificate. If approved, the
#             certificate becomes valid after it's requested.
#             (You may delay the valid date by specifying a non zero
#             number for the value of 'NotBefore',
#             eg. NotBefore=5. That means if the request is approved,
#             the certificate will become valid 5 days after it's
#             requested.)
#
#             These certificates will be stored in LDAP if The O= and
#             OU= suffixes have already been created
#
#
# Other than the user input fields, all other information is hard coded.
#
# User input fields:
# Email - optional
# CommonName - optional
# OrgUnit - optional
# Org - optional
# Street - optional
# Locality - optional
# StateProv - optional
# PostalCode - optional
# Country - optional
# AltEmail - optional
# AltDomain - optional
# AltURI - optional
# AltIPAddr - optional
# PassPhrase - required
# PublicKey - required (This is the PKCS#10 request)
# NotifyEmail - optional
#
# RACF userid/password authentication : not require
# Administrator approval : require
#
#
# =====

```

```

#
<TEMPLATE NAME=5 Year PKI IPSEC Server (Firewall) Certificate>
<TEMPLATE NAME=PKI Server Certificate>
<NICKNAME=5YSIPS>
<CONTENT>
<HTML><HEAD>
<TITLE> Web Based PKIX Certificate Generation Application Pg 2</TITLE>
%%-copyright%%
<SCRIPT LANGUAGE="JavaScript">
<!--
function MissingRequiredFAlert(){
var STRING_MissingRequiredFPrompt=
    "Enter the field value that's not optional."
var STRING_MissingConfirmPwdPrompt=
    "Reenter password."
var STRING_MissingPwdPrompt=
    "Need to enter password before confirm it."
var STRING_UnmatchPwdPrompt=
    "The passwords do not match. Enter again."
var STRING_UnmatchEmailPrompt=
    "The Email addresses for Distinguished name and notification do not match.
Enter again."

if (document.serverform.PassPhrase.value=="")
{
    alert(STRING_MissingRequiredFPrompt);
    document.serverform.PassPhrase.focus();
    return true;
}
else if(document.serverform.ConfirmPassPhrase.value=="") {
alert(STRING_MissingConfirmPwdPrompt);
document.serverform.ConfirmPassPhrase.focus();
return true;
}
else if(document.serverform.PassPhrase.value!=
    document.serverform.ConfirmPassPhrase.value){
alert(STRING_UnmatchPwdPrompt);
document.serverform.ConfirmPassPhrase.focus();
return true;
}
else if(document.serverform.PublicKey.value=="") {
alert(STRING_MissingRequiredFPrompt);
document.serverform.PublicKey.focus();
return true;
}
else if((document.serverform.Email.value != "" &&
    document.serverform.NotifyEmail.value != "") &&
    (document.serverform.Email.value!=
    document.serverform.NotifyEmail.value)){

```

```

alert(String_UnmatchEmailPrompt);
document.serverform.NotifyEmail.focus();
return true;
}
else {
return false;
}
}
//-->
</SCRIPT>

</HEAD>
<BODY>
<H1> 5 Year PKI IPSEC Server (Firewall) Certificate</H1>
<p>
<H2>Choose one of the following:</H2>
<p>
<ul>
<h3><li>Request a New Certificate</h3>
# This ACTION forces userid/pw authentication and runs the task under
# the client's ID
#<FORM NAME=serverform METHOD=POST ACTION=
#           "/PKIServ/ssl-cgi-bin/auth/careq.rexx" onSubmit=

# This ACTION forces userid/pw authentication but runs the task under
# the surrogate ID
#<FORM NAME=serverform METHOD=POST ACTION=
#           "/PKIServ/ssl-cgi-bin/surrogateauth/careq.rexx" onSubmit=

# This ACTION is for non z/OS clients. The task runs under the
# surrogate ID
<FORM NAME=serverform METHOD=POST ACTION=
           "/PKIServ/ssl-cgi-bin/careq.rexx" onSubmit=
           "if(MissingRequiredFAlert()) return false; else return true;">

<INPUT NAME="Template" TYPE="hidden" VALUE="[tplname]">
<p> Enter values for the following field(s)
%%Email (Optional)%%
%%CommonName (Optional)%%
%%OrgUnit (Optional)%%
%%OrgUnit2 (Optional)%%
%%Org (Optional)%%
%%Street (Optional)%%
%%Locality (Optional)%%
%%StateProv (Optional)%%
%%PostalCode (Optional)%%
%%Country (Optional)%%
%%AltEmail (Optional)%%
%%AltDomain (Optional)%%

```

```

%%AltURI (Optional)%%
%%AltIPAddr (Optional)%%
%%Requestor (Optional)%%
%%NotifyEmail (Optional)%%
%%PassPhrase%%
%%PublicKey%%
<p>
<INPUT TYPE="submit" VALUE="Submit certificate request"
ONCLICK="if(MissingRequiredFAlert()) return false; else return true;">
<INPUT TYPE="reset" VALUE="Clear">
</FORM>
<p>
<H3><li>Pick Up a Previously Issued Certificate</H3>

<FORM METHOD=GET ACTION="/PKIServ/ssl-cgi/caretrieve.rexx">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tplname]">
<INPUT TYPE="submit" VALUE="Retrieve your certificate">
</FORM>
</ul>
<p>%%-pagefooter%%
</BODY>
</HTML>
</CONTENT>
<CONSTANT>
%%KeyUsage=handshake%%
%%KeyUsage=dataencrypt%%
%%NotBefore=0%%
%%NotAfter=1825%%
%%SignWith=PKI:%%
</CONSTANT>
<ADMINAPPROVE>
%%CommonName (Optional)%%
%%OrgUnit (Optional)%%
%%OrgUnit (Optional)%%
%%Org (Optional)%%
%%Locality (Optional)%%
%%StateProv (Optional)%%
%%Country (Optional)%%
%%AltEmail (Optional)%%
%%AltDomain (Optional)%%
%%AltURI (Optional)%%
%%AltIPAddr (Optional)%%
%%NotBefore (optional)%%
%%NotAfter (Optional)%%
%%KeyUsage (Optional)%%
</ADMINAPPROVE>
<SUCCESSCONTENT>
%%-requestok%%
</SUCCESSCONTENT>

```

```

<FAILURECONTENT>
  %%-requestbad%%
</FAILURECONTENT>

<RETRIEVECONTENT>
<HTML><HEAD>
<TITLE> Web Based PKIX Certificate Generation Application Pg 3</TITLE>
%%-copyright%%
<SCRIPT LANGUAGE="JavaScript">
<!--
function MissingTransIdAlert(){
var STRING_MissingTransIdPrompt=
    "Enter the transaction ID assigned to the certificate.";
if(document.retrieveform.TransactionId.value==""){
    alert(STRING_MissingTransIdPrompt);
    document.retrieveform.TransactionId.focus();
    return true;
}
else {
    return false;
}
}
//-->
</SCRIPT>
</HEAD>

<BODY>
<H1> Retrieve Your [tmplname]</H1>
<H3>Please bookmark this page</h3>
<p>Since your certificate may not have been issued yet, we recommend
that you create a bookmark to this location so that when you return to
this bookmark, the browser will display your transaction ID.
This is the easiest way to check your status.

# This ACTION forces userid/pw authentication and runs the task
# under the client's ID
#<FORM NAME=retrieveform METHOD=POST ACTION=
#    "/PKIServ/ssl-cgi-bin/auth/cagetcert.rexx" onSubmit=
#
# This ACTION forces userid/pw authentication but runs the task
# under the surrogate ID
#<FORM NAME=retrieveform METHOD=POST ACTION=
#    "/PKIServ/ssl-cgi-bin/surrogateauth/cagetcert.rexx" onSubmit=
#
# This ACTION is for non z/OS clients. The task runs under surrogate ID
<FORM NAME=retrieveform METHOD=POST ACTION=
    "/PKIServ/ssl-cgi-bin/cagetcert.rexx" onSubmit=
    "if(MissingTransIdAlert()) return false; else return true;">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">

```

```

<p> Enter values for the following field(s)
%%TransactionId%%
%%ChallengePassPhrase (optional)%%
<p>
<INPUT TYPE="submit" VALUE="Continue">
</FORM>
<p>%%-pagefooter%%
</BODY>
</HTML>
</RETRIEVECONTENT>
<RETURNCERT>
%%-returnpkcs10cert%%
</RETURNCERT>
</TEMPLATE>

#
# =====
#
# Template Name - 5 Year PKI Intermediate CA Certificate
#
# Function - Creates a 5 year CA certificate. If approved, the
#             certificate becomes valid after it's requested.
#             (You may delay the valid date by specifying a non zero
#             number for the value of 'NotBefore',
#             eg. NotBefore=5. That means if the request is approved,
#             the certificate will become valid 5 days after it's
#             requested.)
#
#             These certificates will be stored in LDAP if The O= and
#             OU= suffixes have already been created
#
#
# Other than the user input fields, all other information is hard coded.
#
# User input fields:
# Email - optional
# CommonName - optional
# OrgUnit - optional
# Org - optional
# Street - optional
# Locality - optional
# StateProv - optional
# PostalCode - optional
# Country - optional
# AltEmail - optional
# AltDomain - optional
# AltURI - optional
# AltIPAddr - optional
# PassPhrase - required

```



```

# PublicKey - required (This is the PKCS#10 request)
# NotifyEmail - optional
#
# RACF userid/password authentication : require
# Administrator approval             : not require
#
#
# =====
#
<TEMPLATE NAME=5 Year PKI Intermediate CA Certificate>
<TEMPLATE NAME=PKI Server Certificate>
<NICKNAME=5YSCA>
<CONTENT>
<HTML><HEAD>
<TITLE> Web Based PKIX Certificate Generation Application Pg 2</TITLE>
%%-copyright%%
<SCRIPT LANGUAGE="JavaScript">
<!--
function MissingRequiredFAlert(){
var STRING_MissingRequiredFPrompt=
    "Enter the field value that's not optional."
var STRING_MissingConfirmPwdPrompt=
    "Reenter password."
var STRING_MissingPwdPrompt=
    "Need to enter password before confirm it."
var STRING_UnmatchPwdPrompt=
    "The passwords do not match. Enter again."
var STRING_UnmatchEmailPrompt=
    "The Email addresses for Distinguished name and notification do not match.
Enter again."

if (document.serverform.PassPhrase.value=="")
{
    alert(STRING_MissingRequiredFPrompt);
    document.serverform.PassPhrase.focus();
    return true;
}
else if(document.serverform.ConfirmPassPhrase.value=="") {
alert(STRING_MissingConfirmPwdPrompt);
document.serverform.ConfirmPassPhrase.focus();
return true;
}
else if(document.serverform.PassPhrase.value!=
    document.serverform.ConfirmPassPhrase.value){
alert(STRING_UnmatchPwdPrompt);
document.serverform.ConfirmPassPhrase.focus();
return true;
}
else if(document.serverform.PublicKey.value=="") {

```

```

alert(String_MissingRequiredFPrompt);
document.serverform.PublicKey.focus();
return true;
}
else if((document.serverform.Email.value != "" &&
        document.serverform.NotifyEmail.value != "") &&
        (document.serverform.Email.value !=
         document.serverform.NotifyEmail.value)){
alert(String_UnmatchEmailPrompt);
document.serverform.NotifyEmail.focus();
return true;
}
else {
return false;
}
}
//-->
</SCRIPT>

</HEAD>
<BODY>
<H1> 5 Year PKI Intermediate CA Certificate</H1>
<p>
<H2>Choose one of the following:</H2>
<p>
<ul>
<h3><li>Request a New Certificate</h3>
# This ACTION forces userid/pw authentication and runs the task under
# the client's ID
<FORM NAME=serverform METHOD=POST ACTION=
        "/PKIServ/ssl-cgi-bin/auth/careq.rexx" onSubmit=

# This ACTION forces userid/pw authentication but runs the task under
# the surrogate ID
#<FORM NAME=serverform METHOD=POST ACTION=
#        "/PKIServ/ssl-cgi-bin/surrogateauth/careq.rexx" onSubmit=

# This ACTION is for non z/OS clients. The task runs under the
# surrogate ID
#<FORM NAME=serverform METHOD=POST ACTION=
#        "/PKIServ/ssl-cgi-bin/careq.rexx" onSubmit=
        "if(MissingRequiredFAlert()) return false; else return true;">

<INPUT NAME="Template" TYPE="hidden" VALUE="[tplname]">
<p> Enter values for the following field(s)
%%Email (Optional)%%
%%CommonName (Optional)%%
%%OrgUnit (Optional)%%
%%OrgUnit2 (Optional)%%

```

```

%%Org (Optional)%%
%%Street (Optional)%%
%%Locality (Optional)%%
%%StateProv (Optional)%%
%%PostalCode (Optional)%%
%%Country (Optional)%%
%%AltEmail (Optional)%%
%%AltDomain (Optional)%%
%%AltURI (Optional)%%
%%AltIPAddr (Optional)%%
%%Requestor (Optional)%%
%%NotifyEmail (Optional)%%
%%PassPhrase%%
%%PublicKey%%
<p>
<INPUT TYPE="submit" VALUE="Submit certificate request"
ONCLICK="if(MissingRequiredFAlert()) return false; else return true;">
<INPUT TYPE="reset" VALUE="Clear">
</FORM>
<p>
<H3><li>Pick Up a Previously Issued Certificate</H3>

<FORM METHOD=GET ACTION="/PKIServ/ssl-cgi/caretrieve.rexx">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
<INPUT TYPE="submit" VALUE="Retrieve your certificate">
</FORM>
</ul>
<p>%%-pagefooter%%
</BODY>
</HTML>
</CONTENT>
<APPL>
%%UserId%%
</APPL>
<CONSTANT>
%%NotBefore=0%%
%%NotAfter=1825%%
%%KeyUsage=certsign%%
%%SignWith=PKI:%%
</CONSTANT>
<SUCCESSCONTENT>
%%-requestok%%
</SUCCESSCONTENT>
<FAILURECONTENT>
%%-requestbad%%
</FAILURECONTENT>

<RETRIEVECONTENT>
<HTML><HEAD>

```

```

<TITLE> Web Based PKIX Certificate Generation Application Pg 3</TITLE>
%%-copyright%%
<SCRIPT LANGUAGE="JavaScript">
<!--
function MissingTransIdAlert(){
var STRING_MissingTransIdPrompt=
    "Enter the transaction ID assigned to the certificate.";
if(document.retrieveform.TransactionId.value==""){
    alert(STRING_MissingTransIdPrompt);
    document.retrieveform.TransactionId.focus();
    return true;
}
else {
    return false;
}
}
//-->
</SCRIPT>
</HEAD>

<BODY>
<H1> Retrieve Your [tmplname]</H1>
<H3>Please bookmark this page</h3>
<p>Since your certificate may not have been issued yet, we recommend
that you create a bookmark to this location so that when you return to
this bookmark, the browser will display your transaction ID.
This is the easiest way to check your status.

# This ACTION forces userid/pw authentication and runs the task
# under the client's ID
<FORM NAME=retrieveform METHOD=POST ACTION=
    "/PKIServ/ssl-cgi-bin/auth/cagetcert.rexx" onSubmit=
#
# This ACTION forces userid/pw authentication but runs the task
# under the surrogate ID
#<FORM NAME=retrieveform METHOD=POST ACTION=
#     "/PKIServ/ssl-cgi-bin/surrogateauth/cagetcert.rexx" onSubmit=
#
# This ACTION is for non z/OS clients. The task runs under surrogate ID
#<FORM NAME=retrieveform METHOD=POST ACTION=
#     "/PKIServ/ssl-cgi-bin/cagetcert.rexx" onSubmit=
#         "if(MissingTransIdAlert()) return false; else return true;">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
<p> Enter values for the following field(s)
%%TransactionId%%
%%ChallengePassPhrase (optional)%%
<p>
<INPUT TYPE="submit" VALUE="Continue">
</FORM>

```

```

<p>%%-pagefooter%%
</BODY>
</HTML>
</RETRIEVECONTENT>
<RETURNCERT>
%%-returnpkcs10cert%%
</RETURNCERT>
</TEMPLATE>

#
# =====
#
# Sample INSERTS
#
# =====
# @D3C
<INSERT NAME=-AdditionalHeadIE>
<OBJECT
  classid="clsid:127698e4-e730-4e5c-a2b1-21490a70c8a1"
  CODEBASE="xenroll.cab#Version=5,131,3659,0"
  id="certmgr"
>
</OBJECT>
</INSERT>

<INSERT NAME=-requestok>
<HTML><HEAD>
<TITLE> Web Based Certificate Generation Success</TITLE>
</HEAD>
<BODY>
<H1> Request submitted successfully</H1>
[errorinfo]
<p> Here's your transaction ID. You will need it to retrieve your
certificate. Press 'Continue' to retrieve the certificate.
<p> <TABLE BORDER><TR><TD>[transactionid]</TD></TR></TABLE>
<FORM METHOD=GET ACTION="/PKIServ/ssl-cgi/caretrieve.rexx">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
<INPUT NAME="TransactionId" TYPE="hidden" VALUE="[transactionid]">
<INPUT TYPE="submit" VALUE="Continue">
</FORM>
<p>%%-pagefooter%%
</BODY>
</HTML>
</INSERT>

<INSERT NAME=-requestbad>
<HTML><HEAD>
<TITLE> Web Based Certificate Generation Failure</TITLE>
</HEAD>

```

```

<BODY>
<H1> Request was not successful</H1>
<p> Please correct the problem or report the error to your Web admin
person<br>
<PRE>
[errorinfo]
</PRE>
<p>%%-pagefooter%%
</BODY>
</HTML>
</INSERT>

```

```

<INSERT NAME=-renewrevokeok>
<HTML><HEAD>
<TITLE> Web Based Certificate Renew/Revoke Success</TITLE>
</HEAD>
<BODY>
<H1> Request submitted successfully</H1>
<FORM METHOD=GET ACTION="/PKIServ/public-cgi/camain.rexx">
<INPUT TYPE="submit" VALUE="Home Page">
</FORM>
<p>%%-pagefooter%%
</BODY>
</HTML>
</INSERT>

```

```

<INSERT NAME=-renewrevokebad>
<HTML><HEAD>
<TITLE> Web Based Certificate Renew/Revoke Failure</TITLE>
</HEAD>
<BODY>
<H1> Request was not successful</H1>
<p> Please correct the problem or report the error to your Web admin
person<br>
<PRE>
[errorinfo]
</PRE>
<FORM METHOD=GET ACTION="/PKIServ/public-cgi/camain.rexx">
<INPUT TYPE="submit" VALUE="Home Page">
</FORM>
<p>%%-pagefooter%%
</BODY>
</HTML>
</INSERT>

```

```

<INSERT NAME=-returnpkcs10cert>
<HTML><HEAD>
<TITLE> Web Based SAF Certificate Generation Application Pg 4</TITLE>
</HEAD>

```

```

<BODY>
<H1> Here's your Certificate. Cut and paste it to a file</H1>
<TABLE BORDER><TR><TD>
<PRE>
[base64cert]
</PRE>
</TD></TR></TABLE>
<p>%%-pagefooter%%
</BODY>
</HTML>
</INSERT>

<INSERT NAME=returnbrowsercertNS>
[base64cert]
</INSERT>

<INSERT NAME=returnbrowsercertIE>
<HTML>
<HEAD>
<TITLE>MSIE Certificate Install</TITLE>
# @D3C
<OBJECT
  classid="clsid:127698e4-e730-4e5c-a2b1-21490a70c8a1"
  CODEBASE="xenroll.cab#Version=5,131,3659,0"
  id="certmgr"
>
</OBJECT>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub INSTALL_OnClick
  Dim pkcs7data, errmsg, rc
  On Error Resume Next
  certmgr.DeleteRequestCert = false
  err.clear
  certmgr.WriteCertToCSP = true
  pkcs7data = "[iecert]"
  certmgr.acceptPKCS7(pkcs7data)
  if err.number <> 0 then
  certmgr.WriteCertToCSP = false
    err.clear
    certmgr.acceptPKCS7(pkcs7data)
  end if
  if err.number <> 0 then
  errmsg = "Your new certificate failed to install. " & _
    "Please ensure that you are using the same browser " & _
    "that you used when making the certificate request. "
  rc = MsgBox (errmsg, 48, "Certificate Installation")

```

```

        else
        errmsg = "Your new certificate installed successfully."
        rc = MsgBox (errmsg, 64, "Certificate Installation")
        end if
    End Sub
    // -->
</SCRIPT>
<h1>Internet Explorer certificate install</h1>
<p> Click &quot;Install Certificate&quot; to store your new
certificate into your browser
<TABLE>
<TR> <br>
<TD><INPUT TYPE="BUTTON" VALUE="Install Certificate" NAME="INSTALL" >
<FORM METHOD=GET ACTION="/PKIServ/public-cgi/camain.rexx">
<INPUT NAME="Template" TYPE="hidden" VALUE="[tmplname]">
<INPUT TYPE="submit" VALUE="Home page">
</FORM>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
</INSERT>
#
# =====
#
# X.509 fields (INSERTs) valid for certificate requests
#
# =====
#
<INSERT NAME=KeyUsage>
<p> Indicate the intended purpose for the certificate [optfield] <BR>
<SELECT NAME="KeyUsage" MULTIPLE>
  <OPTION VALUE="handshake">Protocol handshaking (e.g., SSL)
  <OPTION VALUE="dataencrypt">Data encryption
  <OPTION VALUE="certsign">Certificate signing
  <OPTION VALUE="docsign">Document signing (nonrepudiation)
</SELECT>
</INSERT>

<INSERT NAME=NotBefore>
<p> Number of days after today before the certificate becomes current
[optfield] <BR>
<SELECT NAME="NotBefore">
  <OPTION> 0
  <OPTION> 30
</SELECT>
</INSERT>

```



```

<INSERT NAME=NotAfter>
<p> Length of time that the certificate is current [optfield] <BR>
<SELECT NAME="NotAfter">
  <OPTION value="365">1 Year
  <OPTION value="730">2 Years
</SELECT>
</INSERT>

<INSERT NAME=Country>
<p> Country [optfield] <BR>
<INPUT NAME="Country" TYPE="text" SIZE=2 maxlength="2">
</INSERT>

<INSERT NAME=Org>
<p> Organization [optfield] <BR>
<INPUT NAME="Org" TYPE="text" SIZE=64 maxlength="64">
</INSERT>

<INSERT NAME=OrgUnit>
<p> Organizational Unit [optfield] <BR>
<INPUT NAME="OrgUnit" TYPE="text" SIZE=64 maxlength="64">
</INSERT>

<INSERT NAME=OrgUnit2>
<p> Organizational Unit [optfield] <BR>
<INPUT NAME="OrgUnit2" TYPE="text" SIZE=64 maxlength="64">
</INSERT>

<INSERT NAME=Locality>
<p> Locality [optfield] <BR>
<INPUT NAME="Locality" TYPE="text" SIZE=64 maxlength="64">
</INSERT>

<INSERT NAME=StateProv>
<p> State or Province [optfield] <BR>
<INPUT NAME="StateProv" TYPE="text" SIZE=64 maxlength="64">
</INSERT>

<INSERT NAME=CommonName>
<p> Common Name [optfield] <BR>
<INPUT NAME="CommonName" TYPE="text" SIZE=64 maxlength="64">
</INSERT>

<INSERT NAME=Title>
<p> Title [optfield] <BR>
<INPUT NAME="Title" TYPE="text" SIZE=64 maxlength="64">
</INSERT>

<INSERT NAME=AltIPAddr>

```

```

<p> IP address for alternate name in dotted decimal form [optfield] <BR>
<INPUT NAME="AltIPAddr" TYPE="text" SIZE=15 maxlength="15">
</INSERT>

<INSERT NAME=AltEmail>
<p> Email address for alternate name [optfield] <BR>
<INPUT NAME="AltEmail" TYPE="text" SIZE=100 maxlength="100">
</INSERT>

<INSERT NAME=AltURI>
<p> Uniform Resource Identifier for alternate name [optfield] <BR>
<INPUT NAME="AltURI" TYPE="text" SIZE=100 maxlength="255">
</INSERT>

<INSERT NAME=AltDomain>
<p> Domain name for alternate name [optfield] <BR>
<INPUT NAME="AltDomain" TYPE="text" SIZE=100 maxlength="100">
</INSERT>

<INSERT NAME=Street>
<p> Street address [optfield] <BR>
<INPUT NAME="Street" TYPE="text" MAXLENGTH=64 SIZE=64>
</INSERT>

<INSERT NAME=PostalCode>
<p> Zipcode or postal code [optfield] <BR>
<INPUT NAME="PostalCode" TYPE="text" MAXLENGTH=64 SIZE=64>
</INSERT>

<INSERT NAME=Email>
<p> Email address for distinguished name [optfield] <BR>
<INPUT NAME="Email" TYPE="text" MAXLENGTH=64 SIZE=64>
</INSERT>

<INSERT NAME=SignWith>
<p> Component:/key-Label used to sign this certificate [optfield] <BR>
<p> e.g., "SAF:CERTAUTH/Local CA Cert" sign by CERTAUTH certificate
   "Local CA Cert"
<INPUT NAME="SignWith" TYPE="text" SIZE=45 maxlength="45">
</INSERT>

<INSERT NAME=PublicKey>
<p> Base64 encoded PKCS#10 certificate request [optfield] <BR>
<TEXTAREA NAME="PublicKey"
  COLS="70"
  ROWS="12"
  WRAP="OFF">
</TEXTAREA>
</INSERT>

```

```

<INSERT NAME=PublicKeyNS>
<p> Select a key size
<KEYGEN NAME="PublicKey">
<p>
<INPUT TYPE="Submit" VALUE="Submit certificate request">
</INSERT>

<INSERT NAME=PublicKey2NS>
<p> Select a key size
<KEYGEN NAME="PublicKey">
<p>
<INPUT TYPE="Submit" VALUE="Submit certificate request">
</INSERT>

<INSERT NAME=PublicKeyIE>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub SendReq

    On Error Resume Next
    Dim pkcs10data,DN,i,Message

    DN= ""
    CommonName= "Unspecified Distinguished Name"
    DN= "CN=" + CommonName + ";";
    certmgr.KeySpec = 1
    KeyUsage = "1.3.6.1.5.5.7.3.2"
    i = document.all.CSP.options.selectedIndex
    certmgr.providerName = document.all.CSP.options(i).text
    certmgr.providerType = document.all.CSP.options(i).value
    If document.CertReq.KeyProt.value = 1 Then
        certmgr.GenKeyFlags = 3
    Else
        certmgr.GenKeyFlags = 1
    End If

    pkcs10data = ""
    pkcs10data = certmgr.CreatePKCS10(DN, KeyUsage)
    document.CertReq.PublicKey.value = pkcs10data

    If Len(pkcs10data) > 0 Then
        document.CertReq.submit()
        return True
    Else
        return MsgBox ("PKCS10 Creation Failed",48,"Certificate request")
    End If

```

```

End Sub
// -->
</SCRIPT>

<p> Select the following key information
<p> Cryptographic Service Provider
<select name="CSP">
<script language="VBScript">
    On Error Resume Next
        Dim i, csp, sv

        certmgr.providerType = 1
        i = 0
        csp = ""
        csp = certmgr.enumProviders(i,0)
        sv = "SELECTED"
# @D3C
        If Len(csp) = 0 Then
            errmsg = "Your PC needs a Windows upgrade before certificates " & _
                "can be requested. Click the 'Tools' option on the browser " & _
                "menu then 'Windows Update' to retrieve the upgrade. "
            Call MsgBox(errmsg,8,"Security Warning")
        End If
        While Len(csp) <> 0
            document.write("<OPTION VALUE=1 " & sv & ">" & csp & "</OPTION>")
            i = i + 1
            csp = ""
            csp = certmgr.enumProviders(i,0)
            sv = ""
        Wend
</script>
</select>

<p> Enable strong private key protection?
<select name="KeyProt">
    <option value="1">Yes</option>
    <option value="0" selected>No</option>
</select>
<input type="hidden" name="PublicKey" value="">
<p>
<INPUT TYPE="Button" VALUE="Submit certificate request" ,
    ONCLICK="if (MissingRequiredFAlert()) return false; SendReq()">
</INSERT>

<INSERT NAME=PublicKey2IE>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub SendReq

```

```

On Error Resume Next
Dim pkcs10data,DN,i,Message

    DN= ""
    CommonName= "Unspecified Distinguished Name"
    DN= "CN=" + CommonName + ";"
    certmgr.KeySpec = 1
    KeyUsage = "1.3.6.1.5.5.7.3.2"
    i = document.all.CSP.options.selectedIndex
    certmgr.providerName = document.all.CSP.options(i).text
    certmgr.providerType = document.all.CSP.options(i).value
    If document.CertReq.KeyProt.value = 1 Then
        certmgr.GenKeyFlags = 3
    Else
        certmgr.GenKeyFlags = 1
    End If

    pkcs10data = ""
    pkcs10data = certmgr.CreatePKCS10(DN, KeyUsage)
    document.CertReq.PublicKey.value = pkcs10data

    If Len(pkcs10data) > 0 Then
        document.CertReq.submit()
        return True
    Else
        return MsgBox ("PKCS10 Creation Failed",48,"Certificate request")
    End If

End Sub
// -->
</SCRIPT>

<p> Select the following key information
<p> Cryptographic Service Provider
<select name="CSP">
<script language="VBScript">
    On Error Resume Next
    Dim i, csp, sv

    certmgr.providerType = 1
    i = 0
    csp = ""
    csp = certmgr.enumProviders(i,0)
    sv = "SELECTED"
# @D3C
    If Len(csp) = 0 Then
        errmsg = "Your PC needs a Windows upgrade before certificates " & _

```

```

    "can be requested. Click the 'Tools' option on the browser " & _
    "menu then 'Windows Update' to retrieve the upgrade. "
    Call MsgBox(errmsg,8,"Security Warning")
End If
While Len(csp) <> 0
    document.write("<OPTION VALUE=1 " & sv & ">" & csp & "</OPTION>")
    i = i + 1
    csp = ""
    csp = certmgr.enumProviders(i,0)
    sv = ""
Wend
</script>
</select>

<p> Enable strong private key protection?
<select name="KeyProt">
    <option value="1">Yes</option>
    <option value="0" selected>No</option>
</select>
<input type="hidden" name="PublicKey" value="">
<p>
<INPUT TYPE="Button" VALUE="Submit certificate request" ,
    ONCLICK="if (ValidateEntry()) return false; SendReq()">
</INSERT>

#
# =====
#
# non-X.509 certificate request fields (INSERTs)
#
# =====
#
<INSERT NAME=UserId>
<p> Owning SAF User ID [optfield] <BR>
<INPUT NAME="UserId" TYPE="text" SIZE=8 maxlength="8">
</INSERT>

<INSERT NAME=Label>
<p> Label assigned to certificate being requested [optfield] <BR>
<INPUT NAME="Label" TYPE="text" SIZE=32 maxlength="32">
</INSERT>

<INSERT NAME=Requestor>
<p> Your name for tracking this request [optfield] <BR>
<INPUT NAME="Requestor" TYPE="text" SIZE=32 maxlength="32">
</INSERT>

<INSERT NAME=PassPhrase>
<p> Pass phrase for securing this request. You will need to supply

```

```

this value when retrieving your certificate [optfield] <BR>
<INPUT NAME="PassPhrase" TYPE="password" SIZE=32 maxlength="32"> <BR>
<p> Reenter your pass phrase to confirm <BR>
<INPUT NAME="ConfirmPassPhrase" TYPE="password" SIZE=32
  maxlength="32">
</INSERT>

<INSERT NAME=ChallengePassPhrase>
<p> If you specified a pass phrase when submitting the certificate
request, type it here, exactly as you typed it on the
request form <BR>
<INPUT NAME="ChallengePassPhrase" TYPE="password" SIZE=32
maxlength="32">
</INSERT>

<INSERT NAME=HostIdMap>
<p> HostIdMapping Extension value in subject-id@host-name form
  [optfield] <BR>
<INPUT NAME="HostIdMap" TYPE="text" SIZE=100 maxlength="100">
</INSERT>

<INSERT NAME=TransactionId>
<p> Enter the assigned transaction ID [optfield] <BR>
<INPUT NAME="TransactionId" TYPE="text" SIZE=56 maxlength="56"
VALUE="[transactionid]">
</INSERT>

<INSERT NAME=NotifyEmail>
<p> Email address for notification purposes [optfield] <BR>
<INPUT NAME="NotifyEmail" TYPE="text" SIZE=64 MAXLENGTH="64">
</INSERT>

#####
#
#           Additional section
#
#####

<INSERT NAME=-copyright>
<!--
/*****
/*
/* Licensed Materials - Property of IBM
/* 5694-A01
/* (C) Copyright IBM Corp. 2001
/*
/* *****/
-->
<META HTTP-EQUIV="Content Type" content="text/html; charset=ISO-8859-1">

```

</INSERT>

<INSERT NAME=-pagefooter>

email: webmaster@your-company.com
</INSERT>

PKI Services subcomponents and message levels

Table B-1 PKI Services message subcomponents

Subcomponent	Meaning
*	All subcomponents
CORE	Core PKI functions
DB	VSAM data store activities
LDAP	LDAP posting operations
PKID	PKI Services Daemon
POLICY	CRL policy processing
SAF	Security related

Table B-2 PKI Services message levels

Subcomponent	Meaning
S	Severe
E	Severe and error
W	Severe, error, and warning (default setting)
I	Severe, error, warning, and informational
D	Severe, error, warning, informational, and diagnostic
V	Verbose. (Very verbose - lots of messages!)

JCL samples

Example: B-7 VSAM data set Creation JCL

```
//TRAUNER1 JOB MSGCLASS=H,CLASS=A,NOTIFY=TRAUNER  
//*****
```



```

/** SAMP:      IKYCVSAM                                *
/**                                                    *
/** Licensed Materials - Property of IBM                *
/** 5694-A01                                             *
/** (C) Copyright IBM Corp. 2001, 2002                 *
/** Status = HKY7707                                    *
/**                                                    *
/*******                                              *
/**                                                    *
/** This sample JCL may be used to create the VSAM data sets *
/** PKI Services utilizes to store certificate requests and *
/** issued certificates.                                   *
/**                                                    *
/*******                                              *
/**                                                    *
/** Caution: This is neither a JCL procedure nor a complete job. *
/** Before using this job step, you will have to make the following *
/** modifications:                                         *
/**                                                    *
/** 1) Change the job card to meet your system requirements. *
/**                                                    *
/** 2) If you wish to change the data set qualifiers from the *
/** default value change all occurrences of "TRAUNER.WEBPKI1.VSAM" *
/** to a preferred value. If you choose to modify this value, be *
/** be sure to also modify the sample configuration file *
/** appropriately(/etc/pkiserv/pkiserv.conf). *
/**                                                    *
/** 3) If you are using VSAM record level sharing (RLS), perform *
/** the following steps:                                   *
/**                                                    *
/** a) Replace the VOL(vvvvvv) statements in the DEFKSDS step *
/** with STORCLAS(class-name) where class-name is the name of *
/** the storage class defined for VSAM RLS. *
/**                                                    *
/** b) Remove the VOL(vvvvvv) statements from the DEFALTDX step. *
/**                                                    *
/** c) Remove all the SPANNED and CISIZE statements. *
/**                                                    *
/** If not using VSAM RLS, change all occurrences of vvvvvv to *
/** the VOLSER value appropriate for the system this job is to be *
/** run on. Do not remove the SPANNED and CISIZE statements. *
/**                                                    *
/**                                                    *
/** 4) If you wish to change the default userid to own the VSAM *
/** data set, change the OWNER(TRAUNER.WEBPKI1) operand to the user *
/** want to own the data sets. If you choose to modify this value *
/** ensure you have modified the sample setup REXX exec (IKYSETUP)*
/** to account for this change. *
/**                                                    *

```

```

/** 5) If you wish to change either the primary or secondary record *
/** allocation sizes for either the OST or ICL datasets from the *
/** default value, update the RECORDS(50 50) operands on the *
/** DEFINE CLUSTER or DEFINE ALTERNATE INDEX commands. *
/** *
/** **Note, do not change any of the numeric values other than *
/** CYL or TRK *
/**-----*
/** Change Activity: *
/** *
/** $L1=PKIS3 HKY7707 020314 PDJWS1: VSAM RLS @L1A*
/** $P1=MG00719 HKY7707 020416 PDJWS1: VSAM RLS 2 @P1A*
/** $L2=MG01459 HKY7707 020826 PDJWS1: VSAM scaling - MG01176 @L2A*
/** $P2=MG01459 HKY7707 021022 PDJWS1: JCL errors - MG01346 @P2A*
/** *
/** Change Description: *
/** *
/** C: Added STORCLAS instructions, LOG. Removed VOLUME DDs @L1C*
/** D: Removed FILE(VOLUME) statements @P1A*
/** C: Added more alt indexes and changed allocation parms @L2A*
/** C: Removed VOL keywords from ALTERNATEINDEX statements. @P2A*
/** Removed DD statements from BLDINDEX step. Added @P2A*
/** IEBCGENER step to remove hardcoded binary zeros @P2A*
/**-----*
/** *
/**-----*
/** Delete existing clusters, paths, alt indexes *
/**-----*
//DELCLUST EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE -
    PKI.WEBPKI1.OST -
    CLUSTER -
    PURGE -
    ERASE
DELETE -
    PKI.WEBPKI1.ICL -
    CLUSTER -
    PURGE -
    ERASE
    IF MAXCC LT 9 THEN SET MAXCC = 0
/*
/**-----*
/** Define KSDS *
/**-----*
//DEFKSDS EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *

```

```

DEFINE CLUSTER -
  (NAME(PKI.WEBPKI1.OST) -
  VOL(SBOXA8) -
  RECSZ(1024 32756) -
  INDEXED -
  NOREUSE -
  KEYS(4 0) -
  SHR(2) -
  CYL(3,1) -
  LOG(NONE) -
  OWNER(PKI.PKISRVD) ) -
DATA -
  (NAME(PKI.WEBPKI1.OST.DA) -
  CISZ(1024) -
  SPANNED) -
INDEX -
  (NAME(PKI.WEBPKI1.OST.IX))

DEFINE CLUSTER -
  (NAME(PKI.WEBPKI1.ICL) -
  VOL(SBOXA8) -
  RECSZ(1024 32756) -
  INDEXED -
  NOREUSE -
  KEYS(4 0) -
  SHR(2) -
  CYL(3,1) -
  LOG(NONE) -
  OWNER(PKI.PKISRVD) ) -
DATA -
  (NAME(PKI.WEBPKI1.ICL.DA) -
  CISZ(1024) -
  SPANNED) -
INDEX -
  (NAME(PKI.WEBPKI1.ICL.IX))

/*
/*-----*
/* Repro record of all binary zeros into KSDS *
/*-----*
//MKZEROS EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD *

//SYSUT2 DD DSN=&&GENTMP,UNIT=SYSDA,DISP=(,PASS),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=640),SPACE=(TRK,(1,1))
//SYSIN DD *
GENERATE MAXFLDS=4,MAXLITS=80
RECORD FIELD=(20,X'00000000000000000000000000000000',,1),
        FIELD=(20,X'00000000000000000000000000000000',,21),

```

```

FIELD=(20,X'00000000000000000000000000000000',,41),
FIELD=(20,X'00000000000000000000000000000000',,61)

/*
//REPROKSD EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSDATA DD DSN=*.MKZEROS.SYSUT2,DISP=(OLD,DELETE)
//SYSIN DD *
    REPRO INFILE(SYSDATA) -
        OUTDATASET(PKI.WEBPKI1.OST)
    REPRO INFILE(SYSDATA) -
        OUTDATASET(PKI.WEBPKI1.ICL)
/*
/*-----*
/* Define ALTERNATE INDEX and PATH *
/*-----*
//DEFALTDX EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
    DEFINE ALTERNATEINDEX -
        (NAME(PKI.WEBPKI1.OST.AIX) -
        RELATE(PKI.WEBPKI1.OST)-
        VOL(SBOXA8) -
        TRK(5,1) -
        KEYS(24 44) ) -
    DATA -
        (NAME(PKI.WEBPKI1.OST.AIX.DA)) -
    INDEX -
        (NAME(PKI.WEBPKI1.AIX.IX))
    DEFINE PATH -
        (NAME(PKI.WEBPKI1.OST.PATH) -
        PATHENTRY(PKI.WEBPKI1.OST.AIX))
/*
/*-----*
/* BUILD ALTERNATE INDEX *
/*-----*
//BLDINDEX EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
    BLDINDEX INDATASET(PKI.WEBPKI1.OST) -
        OUTDATASET(PKI.WEBPKI1.OST.AIX)
    BLDINDEX INDATASET(PKI.WEBPKI1.OST) -
        OUTDATASET(PKI.WEBPKI1.OST.STATAIX)
    BLDINDEX INDATASET(PKI.WEBPKI1.ICL) -
        OUTDATASET(PKI.WEBPKI1.ICL.STATAIX)
    BLDINDEX INDATASET(PKI.WEBPKI1.OST) -
        OUTDATASET(PKI.WEBPKI1.OST.REQAIX)
    BLDINDEX INDATASET(PKI.WEBPKI1.ICL) -
        OUTDATASET(PKI.WEBPKI1.ICL.REQAIX)
/*

```

```
/*-----  
/* Print out the cluster  
/*-----  
//PRTCLUST EXEC PGM=IDCAMS  
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *  
    PRINT -  
        INDATASET(PKI.WEBPKI1.OST) CHAR  
    PRINT -  
        INDATASET(PKI.WEBPKI1.ICL) CHAR  
/*
```

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 338. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *z/OS Version 1 Release 3 and 4 Implementation*, SG24-6581
- ▶ *zSeries Crypto Guide Update*, SG24-6870

Other publications

These publications are also relevant as further information sources:

- ▶ *ICSF Administrator's Guide*, SA22-7521
- ▶ *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402
- ▶ *z/OS MVS Initialization and Tuning Reference*, SA22-7592
- ▶ *z/OS Security Server LDAP Server Administration and Use*, SC24-5923
- ▶ *z/OS Security Server PKI Services Guide and Reference*, SA22-7693
- ▶ *z/OS Security Server RACF Callable Services*, SA22-7691
- ▶ *z/OS Security Server RACF Command Language Reference*, SA22-7867
- ▶ *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683
- ▶ *z/OS Security Server RACF System Programmer's Guide*, SA22-7681
- ▶ *z/OS UNIX System Services Command Reference*, SA22-7802
- ▶ *z/OS UNIX System Services Planning*, GA22-7800

Online resources

These Web sites are also relevant as further information sources:

- ▶ Internet Engineering Task Force (IETF)
<http://www.ietf.org>
- ▶ Internet Assigned Numbers Authority (IANA)
<http://www.iana.org>

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Index

A

- Access Control List (ACL) 40
- ACL 40
- ACL types
 - Access 42
 - Directory default 42
 - File default 42
- AIM 50
- AIM stage 3 51
- APPLDATA 55
- application identity mapping 50
- Asymmetric-Keys Master Key 226
- AUTOGID keyword 53
- AUTOID keyword 53

B

- backup_dsn
 - IKYSETUP 184
- base ACL 42
- base64 encoded 90
- base-64 encoding 20
- BER 19
 - encoding rules 19
- BPX.SUPERUSER 50
- bpx_userid
 - IKYSETUP 180
- browser certificates 24

C

- CA certificates 8
 - cross-certification 8
 - recovering profile 91
 - renewing 91
 - using IKYSETUP 176
- ca_dn
 - IKYSETUP 177
- ca_expires
 - IKYSETUP 184
- ca_keysize
 - IKYSETUP 180
- ca_label
 - IKYSETUP 177

- ca_ring

- IKYSETUP 184
- CCF 221
- CERTAUTH 35
- CERTDETAILS 96
- Certificate Policy 5
- Certificate Revocation List *see* CRL
- Certification Authority 4
- Certification Practice Statement 5
- CGI modules 137
- chmod command 50
- CICS 40
- CKDS 232
 - ICSF 232
- CMOS Cryptographic Coprocessor 220
- Code samples
 - httpd.conf sample for PKI Web server 1 266
 - httpd.conf sample for PKI Web server 2 267
 - httpd.envvars sample for the PKI Web server 267
 - IKYCVSAM 331
 - PKI Exit sample 249
 - pkiserv.conf 268
 - pkiserv.envvars 271
 - pkiserv.tmpl 272
- configuring
 - HTTP Web servers 107
 - ICSF 60
 - LDAP for PKI Services 116
 - OCSF and OCEP 127
 - UNIX runtime environment 106
- Coupling Facility 30
- CPACF 228
- CRL 4
 - certification authority 12
 - Delta CRL distribution point 15
 - distribution point 13, 15
 - global CRL 13
 - Renewal phase 4
 - Revocation phase 4
 - revocation policy 12
- CRL distribution point 15
- Cryptographic Services 22
- csfkeys_profile

- IKYSETUP 180
- csfserv_profile
 - IKYSETUP 180
- csfusers_grp
 - IKYSETUP 180

D

- Daemon
 - daemon variable in IKYSETUP 184
 - daemon_uid variable in IKYSETUP 177
- daemon 57
 - IKYSETUP 184
- daemon and server control 57
- daemon_uid
 - IKYSETUP 177
- dataEncipherment 15
- DB2 22
 - data sharing mode 240
 - DBCLI 121
 - DBSPUFI 121
 - DSNAOINI 121
 - key_backup variable in IKYSETUP 181
 - ldap.db2.profile 119
 - load library 57
 - restrict_surrog variable in IKYSETUP 183
 - SYSIBM.SYSCOLUMNS 121
 - TDBM 118
 - TDBM_DB2_DBNAME 120
 - TDBM_DB2_LOCATION 120
 - TDBM_DB2_USERID 120
 - unix_sec variable in IKYSETUP 183
- DER 19
 - encoding rules 19
- DES Master Key 222
- digital certificates 6
 - format 2
 - life cycle 2
 - Authorize fulfillment of request phase 3
 - Fulfillment phase 3
 - Request phase 2
 - Revocation or Renewal phase 3–4
 - Used by Owner phase 3
- digital signatures 15

E

- End Entity 5
- event notification via e-mail 32
- examples

- httpd.envvars definition for PKI Services 113
- httpd.envvars sample for the PKI Web server 267
- IKYCVSAM 331
- IKYSETUP 177
- LDAP procedure 122
- LDAP profile file 120
- PKI Exit sample 249
- pkiserv.conf 268
- pkiserv.envvars 271
- pkiserv.tmpl 272
- EXPORT
 - post-processing 94
- export_dsn
 - IKYSETUP 184
- extended ACL 42
- extended key usage 15

F

- FACILITY class profile
 - IRR.DIGTCERT.ADDRING 81
- FACILITY class profile 63
 - BPX.DAEMON 76
 - BPX.SERVER 76
 - IRR.DIGTCERT 63
 - IRR.DIGTCERT.CONNECT 81
 - IRR.DIGTCERT.GENCERT 76
 - IRR.RPKISERV.PKIADMIN 97
- file security packet 41
- Five Year PKI Intermediate CA Certificate 277
- Five Year PKI IPSEC Server (Firewall) Certificate 277
- Five Year PKI SSL Server Certificate 277
- FSP 41

G

- G4 219
- G5 219
- G6 219
- GENCERT
 - post-processing 94
- GENRENEW
 - post-processing 94
- getfacl command 42
- groups for PKI Services 61

H

HFS 31
 POSIX permission 41
HIGHTRUST 24
HostIdMapping
 Administering Certificates 85
HostIdMappings 23
HTTP/HTTPS 22

I

IANA 338
IBM 4758 Model 2 224
IBM HTTP Server 234
ICL 28
 iclview 88
 vosview 88
ICSF 23, 60
 CKDS 232
 PKDS 232
IETF 4, 338
IKYMV SAM 30
IKYSETUP 37
 backup_dsn 184
 bpx_userid 180
 ca_dn 177
 ca_expires 184
 ca_keysize 180
 ca_label 177
 ca_ring 184
 Compulsory changes 177
 csfkeys_profile 180
 csfserv_profile 180
 csfusers_grp 180
 daemon 184
 daemon_uid 177
 export_dsn 184
 key_backup 181
 log_dsn 184
 pgmcntl_dsn 182
 pki_gid 177
 pkigroup 184
 pkigroup_mem 178
 restrict_surrog 183
REXX 175
surrog 185
surrog_uid 178
unix_sec 183
use_icsf 183

Variables 176
vsamhlq 185
web_dn 179
web_expires 185
web_label 185
web_ring 179
webserver 185
INSERTS section 319
 AdditionalHeadIE 319
 AltDomain 324
 AltEmail 324
 AltIPAddr 323
 AltURI 324
 ChallengePassPhrase 329
 CommonName 323
 Country 323
 Email 324
 HostIdMap 329
 KeyUsage 322
 Label 328
 Locality 323
 NotAfter 323
 NotBefore 322
 NotifyEmail 329
 Org 323
 OrgUnit 323
 OrgUnit2 323
 pagefooter 330
 PassPhrase 328
 PostalCode 324
 PublicKey 324
 PublicKey2IE 326
 PublicKey2NS 325
 PublicKeyIE 325
 PublicKeyNS 325
 renewrevokebad 320
 renewrevokeeok 320
 requestbad 319
 requestok 319
 Requestor 328
 returnbrowsercertIE 321
 returnbrowsercertNS 321
 returnpkcs10cert 320
 SignWith 324
 StateProv 323
 Street 324
 Title 323
 TransactionId 329
 UserId 328

IRRGMAP class 51
IRRIRA00 conversion utility 50
IRRIRA00 utility 50
IRRUMAP class 51

J

JCL
 IKYCVSAM 331
 LDAP procedure 122
Job Role groups 39

K

key ring 68, 108
key_backup
 IKYSETUP 181
keyAgreement 15
keyCertSign 15
keyEncipherment 15
keyfile 109

L

LDAP 6, 59
 ldapcnf utility 119
 LDIF format 242
 Primary Server 241
 Replica Server 243
 Server Replication 240
 TDBM utilities 243
 using encrypted passwords 97
 z/OS 240
LDAP directory 22
 TDBM back-end 22
LDAP password encryption 33
LDAPBIND
 class profile 98
log_dsn
 IKYSETUP 184

M

Makefile.pkiexit 206
Mapping
 host identity 85
Mapping policies 16
MD5 35
message levels
 PKISERV_MSG_LEVEL 272
 PKISERV_MSG_LOGGING 272

MODIFYCERTS 96
MODIFYREQS 96
multiserver 240

N

NotifyEmail 32
NOTRUST 34
NumServers (parameter in pki.conf) 131

O

Object Identifier 18
OCEP 23, 59
 RACF 59
OCSF 22, 59
 RACF 59
OCSF/OCEP
 Configuration to work with PKI Services 127
 variables directory 107
OCSP 6
OMVS 242
One Year PKI S/MIME Browser Certificate 277
One Year PKI SSL Browser Certificate 277
One Year SAF Browser Certificate 277
One Year SAF Server Certificate 277

P

Parallel Sysplex 240
PCI Cryptographic Coprocessor 35
PCICA
 z990 230
PICCC
 Asymmetric-Keys Master Key 226
 FCV 224
 key generation 35
 Symmetric-Keys Master Key 226
PCIXCC) 228
pgmcntl_dsn
 IKYSETUP 182
PKA Key Management Master Key 222
PKA Signature Master Key 222
PKCS#10 6, 25
PKDS 232
 ICSF 232
 sysplex environment 237
PKI
 RACF 61
 template customization 132

- PKI administrator group 61
- PKI data set profiles 63
- PKI Exit 205
 - compilation 207
 - installation and customization 206
 - Main routine 206
 - PKIServ CGIs 205
- PKI issued certificate list (ICL) 107
- PKI object store (OST) 107
- PKI S/MIME browser certificate 23
- PKI Services 21, 32–33, 62
 - creating the key ring 68
 - link from home page 162
 - PKI Services 20
 - template file 137
 - user interface 137
 - variable directory 107
- PKI SSL Server certificate 24
- pki_gid
 - IKYSETUP 177
- pkigroup
 - IKYSETUP 184
- pkigroup_mem
 - IKYSETUP 178
- PKISERV 21
- pkiserv.conf 268
 - parameters
 - AuthName1 98
 - AuthPwd1=LDAPADMIN 98
 - BindProfile1=LDAPKI 98
 - CPS1 162
 - CreateInterval 270
 - CreateOUValue 131
 - CRLDistName=CRL 270
 - CRLDistSize 270
 - CRLDuration 161, 270
 - ExpireWarningTime 270
 - ExpiringMessageForm 271
 - ICLDSN 269
 - ICLRequestorDSN 269
 - ICLStatusDSN 269
 - InitialThreadCount 270
 - KeyRing 131
 - MyPolicy 269
 - ObjectDSN 269
 - ObjectRequestorDSN 269
 - ObjectStatusDSN 269
 - ObjectTidDSN 269
 - Policy1Notice1 162
 - Policy1Notice2 162
 - Policy1Org 162
 - PolicyCritical 161
 - PolicyName1 162
 - PolicyRequired 161
 - PostInterval 131
 - ReadyMessageForm 270
 - RejectMessageForm 271
 - RemoveCompletedReqs 269
 - RemoveExpiredCerts 270
 - RemoveInactiveReqs 269
 - RetryMissingSuffix 131
 - Server1 131
 - SharedVSAM 270
 - SigAlg1 161
 - TimeBetweenCRLs 161
 - UserNoticeText1 162
- pkiserv.envvars 271
- pkiserv.tmpl 137
- PKITP 128
 - setup 128
- pkitp IVP 128
- pkitp.so 128
- PKIX
 - schema 22
 - standards 4
- Policies Mappings 16
- post-processing
 - EXPORT 94
 - GENRENEW 94
 - REVOKE 95
- profile 95
 - FACILITY class 95
 - IRR.DIGTCERT.ADD 96
 - IRR.DIGTCERT.EXPORT 96
 - IRR.DIGTCERT.GENCERT 96
 - IRR.DIGTCERT.GENRENEW 96
 - IRR.DIGTCERT.REQCERT 96
 - IRR.DIGTCERT.REQRENEW 96
 - IRR.DIGTCERT.REVOKE 96
 - IRR.DIGTCERT.VERIFY 96
 - LDAPBIND class profile 98
- program control 56
- psot-processing
 - GENCERT 94
 - REQCERT 95
- Public Key Certificate 4

Q

QUERCERTS 96
QUERYREQS 97

R

RACDCERT 37
 DIGTCERT 65
 DIGTNMAP 65
 DIGTRING 65
RACF 23, 58–59
 creating certificates 64
 daemon and server control 57
 OCEP 59
 OCSF 59
 PKI Services 61
 program control 56
 UNIXPRIV class 43
 Web server 58
RACF database
 AIM 50
RACF default group 39
 STC 39
 STCG 39
 STCGRP 39
RACF group structure 38
Redbooks Web site 338
 Contact us xi
Registration Authority 5
REQCERT
 post-processing 95
REQDETAILS 97
restrict_surrog
 IKYSETUP 183
REVOKE
 post-processing 95

S

S/390 219
S/MIME 23
 certificate description 294
 fields of certificate 294
SAF
 browser certificate 23
 EXPORT 95
 GENCERT 95
 server certificate 23
 userID 23
SAF callable services 21

SEARCH command 50
Security Server 20
sendmail utility 32
server certificates 26
setfact command 42, 50
SHA-1 35
SHARED keyword 52
SHARED.IDS profile 52
SSL/TLS 7
SUPERUSER.FILESYS.CHANGEPERMS 42
surrog
 IKYSETUP 185
surrog_uid
 IKYSETUP 178
surrogate user ID 62
Surrogate userid 62
Symmetric-Keys Master Key 226

T

TEMPLATE section of pkiserv.tmpl 161
 ADMINAPPROVE subsection 292
 APPL subsection 301
 CONSTANT subsection 302
 CONTENT subsection 304
 FAILURECONTENT subsection 307
 RETRIEVECONTENT subsection 307
 RETURNCERT subsection 309
 SUCCESSCONTENT subsection 312
templates file 27
TKE 220
TRUST 34
Two Year PKI Browser Certificate For Authenticating 277

U

UID/GID 50
 Automatic assignment 50
UNIX 242
unix_sec
 IKYSETUP 183
UNIXMAP class 50–51
UNIXPRIV class 52
 SUPERUSER.FILESYS.CHANGEPERMS profile 50
use_icsf
 IKYSETUP 183

V

VeriSign 108
VLF 51
VSAM data sets 28
 IKYCVSAM 126
 PKI.WEBPKI1.ICL 88
 PKI.WEBPKI1.OST 88
 VSAM RLS 30
VSAM RLS 30
vsamhq
 IKYSETUP 185

W

Web server 58
Web server certificates 26
web_dn
 IKYSETUP 179
web_expires
 IKYSETUP 185
web_label
 IKYSETUP 185
web_ring
 IKYSETUP 179
webserver
 IKYSETUP 185

X

X.509
 certificate extension fields 14
X.509 Certificate 9

Z

z/OS 40, 108
 LDAP 240
z/OS PKI Services architecture 26
z/OS UNIX level security 56
z/OS V1R4 29
 e-mail notification 29
 POSTALCODE distinguished name qualifiers
 29
 remove clear text LDAP passwords 29
 Support MAIL 29
z990 219
 PCICA 230



Implementing PKI Services on z/OS

(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages



Redbooks

Implementing PKI Services on z/OS

Installation of PKI and all of its prerequisites on z/OS

An example of the PKI Exit

PKI's use of ICSF to store Master Key

This IBM Redbook is designed to help you understand and implement Public Key Infrastructure (PKI) Services on z/OS.

As an overview of PKI Services on z/OS, this book looks into two different ways to install PKI Services: the traditional step-by-step configuration that you do yourself, and by using the new IKYSETUP configuration utility, which enables you to enter certain parameters and then run IKEYSETUP to configure PKI Services for you.

We also look at the PKI Exit and show an example. A chapter about PKI Services and the Cryptographic Coprocessor discusses the Cryptographic solution on the IBM @server zSeries 990 and shows how to boost your SSL connection with hardware encryption.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks